E7

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号
特開2002－222081
（P2002－222081A）
(43)公開日　平成14年８月９日(2002.8.9)

(51)Int.Cl.'	識別記号	ＦＩ	ﾃｰﾏｺｰﾄﾞ(参考)
Ｇ０６Ｆ　9/44		Ｇ０６Ｆ　9/06	６２０Ａ　５Ｂ０７６

審査請求　未請求　請求項の数13　ＯＬ　（全 18 頁）

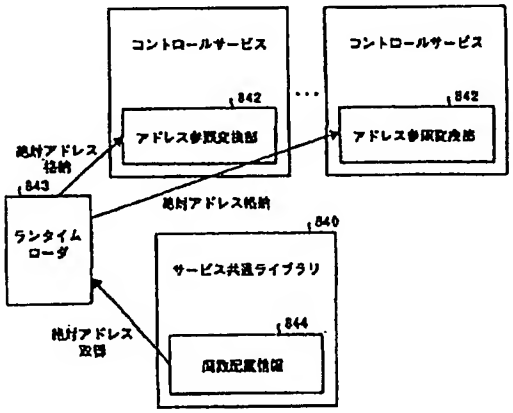
(21)出願番号　特願2001－280642(P2001－280642)
(22)出願日　平成13年９月14日(2001.9.14)
(31)優先権主張番号　特願2000－356699(P2000－356699)
(32)優先日　平成12年11月22日(2000.11.22)
(33)優先権主張国　日本（ＪＰ）

(71)出願人　000006747
株式会社リコー
東京都大田区中馬込１丁目３番６号
(72)発明者　入野　祥明
東京都大田区中馬込１丁目３番６号　株式会社リコー内
(74)代理人　100089118
弁理士　酒井　宏明

Ｆターム(参考)　5B076 AB03 AB04 AB10 AB13 AB15 AB20 AC03 DA06 DB01 DB04 DB09

(54)【発明の名称】　プログラム作成装置、プログラム作成方法、その方法をコンピュータに実行させるプログラム、画像形成装置およびアドレス解決方法

(57)【要約】
【課題】　各モジュールごとに別個に開発することを容易にして作業効率の向上を図るとともに、プログラムサイズの低減を図ること。
【解決手段】　複数のコントロールサービスから共通に呼び出される一または複数の関数と、各関数のメモリ上での絶対アドレスを示す関数配置情報８４４を含むサービス共通ライブラリ８４０を備え、コントロールサービスは、ランタイムローダ８４３によって呼び出す関数の絶対アドレスが実行時に格納されるアドレス参照変換部８４２を有している。



1

【特許請求の範囲】

【請求項１】 コンピュータに実行させるプログラムを作成するプログラム作成装置において、
前記プログラムが参照する外部関数モジュールへのジャンプ命令からなるアドレス参照変換モジュールを作成する作成手段と、
前記プログラムをコンピュータで実行可能な形式に変換する変換手段と、
前記変換手段により変換されたプログラムに、前記作成手段により作成された、当該プログラムが参照する外部関数モジュールへのジャンプ命令からなるアドレス参照変換モジュールを結合する結合手段と、
を備えたことを特徴とするプログラム作成装置。

【請求項２】 さらに、前記プログラムの実行時に前記結合手段により結合されたアドレス参照変換モジュールのジャンプ先の外部関数モジュールのアドレスを検索する検索手段と、
前記検索手段により検索されたアドレスを前記結合手段により結合されたアドレス参照変換モジュールに設定する設定手段と、
を備えたことを特徴とする前記請求項１に記載のプログラム作成装置。

【請求項３】 コンピュータに実行させるプログラムを作成するプログラム作成方法において、
前記プログラムが参照する外部関数モジュールへのジャンプ命令からなるアドレス参照変換モジュールを作成する作成工程と、
前記プログラムをコンピュータで実行可能な形式に変換する変換工程と、
前記変換工程で変換されたプログラムに、前記作成工程で作成された、当該プログラムが参照する外部関数モジュールへのジャンプ命令からなるアドレス参照変換モジュールを結合する結合工程と、
を含んだことを特徴とするプログラム作成方法。

【請求項４】 さらに、前記プログラムの実行時に前記結合工程で結合されたアドレス参照変換モジュールのジャンプ先の外部関数モジュールのアドレスを検索する検索工程と、
前記検索工程で検索されたアドレスを前記結合工程で結合されたアドレス参照変換モジュールに設定する設定工程と、
を含んだことを特徴とする前記請求項３に記載のプログラム作成方法。

【請求項５】 前記請求項３または請求項４に記載された方法をコンピュータに実行させるためのプログラム。

【請求項６】 画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションと、前記アプリケーションと前記ハードウェア資源との間に介在し、ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行う複数のコントロールサービスを含むプラットホームとを備えた画像形成装置であって、
複数の前記アプリケーションから共通に呼び出される一または複数の関数と、各関数のメモリ上での絶対アドレスを示す関数配置情報を含むアプリ共通ライブラリを備え、
前記アプリケーションは、呼び出す関数の絶対アドレスが実行時に格納されるアドレス参照変換部を有することを特徴とする画像形成装置。

【請求項７】 前記複数のコントロールサービスから共通に呼び出される一または複数の関数と、各関数のメモリ上での絶対アドレスを示す関数配置情報とを有するサービス共通ライブラリをさらに備え、
前記コントロールサービスは、呼び出す関数の絶対アドレスが実行時に格納されるアドレス参照変換部を有することを特徴とする請求項６に記載の画像形成装置。

【請求項８】 前記サービス共通ライブラリは、前記アプリ共通ライブラリから呼び出される一または複数の関数と、各関数のメモリ上での絶対アドレスを示す関数配置情報とを有し、
前記アプリ共通ライブラリは、呼び出す関数の絶対アドレスが実行時に格納されるアドレス参照変換部を有することを特徴とする請求項７に記載の画像形成装置。

【請求項９】 画像形成装置の起動時に、前記関数配置情報に基づいて、前記呼び出す関数の絶対アドレスを前記アドレス参照変換部に格納するアドレス参照解決手段をさらに備えたことを特徴とする請求項６～８のいずれか一つに記載の画像形成装置。

【請求項１０】 前記アドレス参照解決手段は、前記サービス共通ライブラリまたは前記アプリ共通ライブラリの先頭の絶対アドレスと該先頭の絶対アドレスからの関数のオフセットを前記アドレス参照変換部に格納することを特徴とする請求項９に記載の画像形成装置。

【請求項１１】 前記コントロールサービスおよび前記アプリケーションは、一または複数のスレッドを含むプロセスとして動作し、
前記サービス共通ライブラリは、前記スレッドに関する処理を実行する関数を含んでいることを特徴とする請求項６～１０のいずれか一つに記載の画像形成装置。

【請求項１２】 画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションと、前記アプリケーションと前記ハードウェア資源との間に介在し、ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行う複数のコントロールサービスを含むプラットホームと、
複数の前記アプリケーションから共通に呼び出される一

2

または複数の関数と各関数のメモリ上での絶対アドレスを示す関数配置情報とを含むアプリ共通ライブラリと、前記複数のコントロールサービスから共通に呼び出される一または複数の関数と各関数のメモリ上での絶対アドレスを示す関数配置情報とを有するサービス共通ライブラリとを備えた画像形成装置を起動する際に、前記コントロールサービスおよび前記アプリケーションから呼び出される関数のアドレスを解決するアドレス解決方法において、

前記呼び出す関数のメモリ上での絶対アドレスを、前記関数配置情報に基づいて、前記アプリケーションおよび前記コントロールサービスのアドレス参照変換部に格納するアドレス参照解決ステップを含んだことを特徴とするアドレス解決方法。

【請求項１３】　前記アドレス参照解決ステップは、前記サービス共通ライブラリまたは前記アプリ共通ライブラリの先頭の絶対アドレスと該先頭の絶対アドレスからの関数のオフセットを前記アドレス参照変換部に格納することを特徴とする請求項１２に記載のアドレス解決方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】この発明は、コンピュータに実行させるプログラムを作成するプログラム作成装置、プログラム作成方法、その方法をコンピュータに実行させるプログラム、プリンタ、コピーまたはファクシミリなどの画像形成処理にかかるユーザサービスを提供する際に動作するコントロールサービスやアプリケーションを共通のライブラリとリンクして生成した画像形成装置およびコントロールサービスやアプリケーションから呼び出される関数のアドレス解決方法に関する。

【０００２】

【従来の技術】近年、プリンタ、コピー、ファクシミリ、スキャナなどの各装置の機能を１つの筐体内に収納した画像形成装置（以下、「複合機」という。）が一般的に知られている。この複合機は、１つの筐体内に表示部、印刷部および撮像部などを設けるとともに、プリンタ、コピーおよびファクシミリ装置にそれぞれ対応する３種類のソフトウェアを設け、ソフトウェアの切り替えによって、当該装置をプリンタ、コピー、スキャナまたはファクシミリ装置として動作させるものである。

【０００３】このような従来の複合機では、内部にプリンタ、コピー、スキャナおよびファクシミリ装置に対応するソフトウェア（汎用ＯＳを含む）をそれぞれ別個に設ける構成となっており、各ソフトウェアの開発に多大の時間を要する。しかも、このような複合機で動作するアプリケーションソフトウェアは、複合機開発元の一カ所で開発を行うことが一般的であった。

【０００４】このため、出願人は、表示部、印刷部および撮像部などの画像形成処理で使用されるハードウェア資源を有し、プリンタ、コピーまたはファクシミリなどの各ユーザサービスにそれぞれ固有の処理を行うアプリケーションを複数搭載し、これらのアプリケーションとハードウェア資源との間に介在して、ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とするハードウェア資源の管理、実行制御並びに画像形成処理を行う各種コントロールサービスからなるプラットホームを備えた画像形成装置を発明した。この画像形成装置によれば、アプリケーションの少なくとも２つが共通的に必要とするハードウェア資源の管理、実行制御並びに画像形成処理を行うプラットホームを備えた構成とすることによって、ソフトウェア開発の効率化を図るとともに、装置全体としての生産性を向上させることが可能となる。

【０００５】このような複合機で動作するソフトウェアを開発する場合、複数のモジュールから共通して呼び出される外部関数などを共通ライブラリとして共有にしておき、開発対象のモジュールのソースファイルからオブジェクトファイルを生成する際に、当該モジュールと外部関数モジュールを結合（リンク）してモジュールを完成することが開発作業効率およびプログラムサイズの低減を図る上で好ましい。

【０００６】このような開発対象のモジュールと外部関数モジュールとを結合する方式としては、従来から以下の（１）スタティックリンク（実行前結合）方式および（２）ダイナミックリンク（実行時結合）方式が一般的に知られている。

【０００７】（１）スタティックリンク（実行前結合）方式：これは、必要となる可能性のあるモジュールを、すべて実行前にあらかじめ結合しておく方式である。このスタティックリンク方式では、開発対象のモジュール側で共通ライブラリの中の関数の仮想メモリ上での絶対アドレスが直接参照される。モジュールがあらかじめ結合されている。したがって、必要が生じたときに必要なモジュールを迅速に実行することができるという利点がある。

【０００８】（２）ダイナミックリンク（実行時結合）方式

これは、実行時に、必要の生じたモジュールを随時結合する方式である。同一のモジュールを複数のプログラムから参照することができる。そのため、共通に利用するモジュールのライブラリを用意することで、プログラム全体のサイズを小さくできるという利点がある。

【０００９】

【発明が解決しようとする課題】しかしながら、上記（１）スタティックリンク（実行前結合）方式、（２）ダイナミックリンク（実行時結合）方式ともに、複合機で動作するソフトウェアを開発する場合には、上記の利点に対して下記のような技術的課題を有している。

【００１０】まず、スタティックリンク（実行前結合）

方式では、すべてのモジュールをあらかじめ実行プログラムに結合しておくため、プログラム全体のサイズが増大するという問題がある。

【００１１】また、上記の複合機では、複数のアプリケーションと、アプリケーションの少なくとも２つが共通的に必要とするサービスを提供する複数のコントロールサービスを有するという各モジュールが独立した構成となっているため、各アプリケーションごとあるいは各コントロールサービスごとに開発を別個に行えることが好ましい。たとえば、コントロールサービスのモジュールの開発を複合機の開発元で行う一方、アプリケーションの開発をサードベンダなどの第三者に委託するという開発方法も考えられる。

【００１２】しかしながら、スタティックリンク方式を採用した場合には、開発対象のモジュール側で外部関数の仮想メモリ上での絶対アドレスが直接参照されるように開発を行う必要があるため、各モジュールを別個に作成することが困難になるという問題がある。特に、共通ライブラリの外部関数の仕様を変更したり、新たな外部関数を追加するような複合機のバージョンアップを行う場合には、共通ライブラリだけでなく、共通ライブラリを呼び出す各アプリケーションや各コントロールサービスのモジュールも、関数の絶対アドレスの変更が必要となるので作業効率が悪いという問題がある。

【００１３】また、ダイナミックリンク（実行時結合）方式では、モジュールの結合処理にともなうオーバヘッドがあるために、実行プログラムの性能が低下しやすいという問題と、実行時にモジュールの結合処理を行なうために、アドレス解決のための処理をモジュールごとに組み込んでおかなければならず、スタティックリンク方式のモジュールと比較してダイナミックリンク方式のモジュールの方がサイズが肥大するという問題があった。

【００１４】この点、たとえば特開平６－２５０８２８号公報では、結合するモジュールを利用頻度の高低によって分類しておき、利用頻度の高いモジュール群はあらかじめ実行プログラムにスタティックリンクしておき、利用頻度の低いモジュール群は実行時にダイナミックリンクするようにすることで、プログラムの実行速度の向上をはかっている。

【００１５】しかし、とくに組み込み型の実行プログラムでは外部からの入力事象によって処理内容が変化するために、モジュールの利用頻度をあらかじめ予想することは困難であるという問題がある。また、上記によっても実行時に結合するモジュールのサイズが肥大するという問題は解決されていない。

【００１６】この発明は上記従来技術による問題点を解決するため、画像形成装置の組み込み型の実行プログラムについて、各モジュールごとに別個に開発することを容易にして作業効率の向上を図るとともに、プログラムサイズの低減を図ることができ、かつ各モジュールの更新・修正などにも柔軟に対応することが可能なプログラム作成装置、プログラム作成方法、その方法をコンピュータに実行させるプログラム、画像形成装置およびアドレス解決方法を提供することを目的とする。

【００１７】

【課題を解決するための手段】上述した課題を解決し、目的を達成するため、請求項１に記載の発明にかかるプログラム作成装置は、コンピュータに実行させるプログラムを作成するプログラム作成装置において、前記プログラムが参照する外部関数モジュールへのジャンプ命令からなるアドレス参照変換モジュールを作成する作成手段と、前記プログラムをコンピュータで実行可能な形式に変換する変換手段と、前記変換手段により変換されたプログラムに、前記作成手段により作成された、当該プログラムが参照する外部関数モジュールへのジャンプ命令からなるアドレス参照変換モジュールを結合する結合手段と、を備えたことを特徴とする。

【００１８】この請求項１に記載の発明によれば、プログラムから参照される外部関数モジュールは、当該モジュールに代えて上記プログラムにスタティックリンクされた、当該モジュールへのジャンプ命令からなるアドレス参照変換モジュールを経由して実行されることになる。

【００１９】また、請求項２に記載の発明にかかるプログラム作成装置は、前記請求項１に記載の発明において、さらに、前記プログラムの実行時に前記結合手段により結合されたアドレス参照変換モジュールのジャンプ先の外部関数モジュールのアドレスを検索する検索手段と、前記検索手段により検索されたアドレスを前記結合手段により結合されたアドレス参照変換モジュールに設定する設定手段と、を備えたことを特徴とする。

【００２０】この請求項２に記載の発明によれば、アドレス参照変換モジュールのジャンプ先のアドレスは、当該モジュールがスタティックリンクされたプログラムの実行時に確定される。

【００２１】また、請求項３に記載の発明にかかるプログラム作成方法は、コンピュータに実行させるプログラムを作成するプログラム作成方法において、前記プログラムが参照する外部関数モジュールへのジャンプ命令からなるアドレス参照変換モジュールを作成する作成工程と、前記プログラムをコンピュータで実行可能な形式に変換する変換工程と、前記変換工程で変換されたプログラムに、前記作成工程で作成された、当該プログラムが参照する外部関数モジュールへのジャンプ命令からなるアドレス参照変換モジュールを結合する結合工程と、を含んだことを特徴とする。

【００２２】この請求項３に記載の発明によれば、プログラムから参照される外部関数モジュールは、当該モジュールに代えて上記プログラムにスタティックリンクされた、当該モジュールへのジャンプ命令からなるアドレ

ス参照変換モジュールを経由して実行されることになる。

【００２３】また、請求項４に記載の発明にかかるプログラム作成方法は、前記請求項３に記載の発明において、さらに、前記結合工程で結合されたアドレス参照変換モジュールのジャンプ先の外部関数モジュールのアドレスを検索する検索工程と、前記検索工程で検索されたアドレスを前記結合工程で結合されたアドレス参照変換モジュールに設定する設定工程と、を含んだことを特徴とする。

【００２４】この請求項４に記載の発明によれば、アドレス参照変換モジュールのジャンプ先のアドレスは、当該モジュールがスタティックリンクされたプログラムの実行時に確定される。

【００２５】また、請求項５に記載の発明は、前記請求項３または請求項４に記載された方法をコンピュータに実行させるプログラムであるので、請求項３または請求項４のいずれか一つの動作をコンピュータによって実行することができる。

【００２６】また、請求項６に記載の発明にかかる画像形成装置は、画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションと、前記アプリケーションと前記ハードウェア資源との間に介在し、ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行う複数のコントロールサービスを含むプラットホームとを備えた画像形成装置であって、複数の前記アプリケーションから共通に呼び出される一または複数の関数と、各関数のメモリ上での絶対アドレスを示す関数配置情報を含むアプリ共通ライブラリを備え、前記アプリケーションは、呼び出す関数の絶対アドレスが実行時に格納されるアドレス参照変換部を有することを特徴とする。

【００２７】この請求項６に記載の発明によれば、複数のアプリケーションから共通に呼び出される一または複数の関数と、各関数のメモリ上での絶対アドレスを示す関数配置情報を含むアプリ共通ライブラリを備えており、アプリケーションでは呼び出す関数の絶対アドレスが実行時に格納されるアドレス参照変換部を有しているので、アプリケーションにアドレス参照変換部を設ければ、呼び出す関数の絶対アドレスを意識した開発を行う必要がなくアプリケーションの開発をアプリ共通ライブラリや他のアプリケーションと別個に行うことが容易となる。また、アプリケーションの開発はアドレス参照変換部を設けることで呼び出す関数の絶対アドレスを開発時に決定する必要がないので、アプリケーションのバージョンアップを容易に行える。従って、アプリケーションの開発の作業効率を向上させることができる。

【００２８】また、請求項７にかかる発明は、請求項６に記載の画像形成装置において、前記複数のコントロールサービスから共通に呼び出される一または複数の関数と、各関数のメモリ上での絶対アドレスを示す関数配置情報とを有するサービス共通ライブラリをさらに備え、前記コントロールサービスは、呼び出す関数の絶対アドレスが実行時に格納されるアドレス参照変換部を有することを特徴とする。

【００２９】この請求項７に記載の発明によれば、複数のコントロールサービスから共通に呼び出される一または複数の関数と、各関数のメモリ上での絶対アドレスを示す関数配置情報とを有するサービス共通ライブラリをさらに備え、前記コントロールサービスは、呼び出す関数の絶対アドレスが実行時に格納されるアドレス参照変換部を有することで、コントロールサービスにアドレス参照変換部を設ければ、呼び出す関数の絶対アドレスを意識した開発を行う必要がなく、コントロールサービスの開発をサービス共通ライブラリや他のコントロールサービスと別個に行うことが容易となる。また、コントロールサービスの開発はアドレス参照変換部を設けることで呼び出す関数の絶対アドレスを開発時に決定する必要がないので、コントロールサービスのバージョンアップを容易に行える。従って、コントロールサービスの開発の作業効率を向上させることができる。

【００３０】また、請求項８にかかる発明は、請求項７に記載の画像形成装置において、前記サービス共通ライブラリは、前記アプリ共通ライブラリから呼び出される一または複数の関数と、各関数のメモリ上での絶対アドレスを示す関数配置情報とを有し、前記アプリ共通ライブラリは、呼び出す関数の絶対アドレスが実行時に格納されるアドレス参照変換部を有することを特徴とする。

【００３１】この請求項８に記載の発明によれば、サービス共通ライブラリはさらにアプリ共通ライブラリから呼び出される一または複数の関数と各関数のメモリ上での絶対アドレスを示す関数配置情報とを有し、アプリ共通ライブラリは呼び出す関数の絶対アドレスが実行時に格納されるアドレス参照変換部を有することで、アプリ共通ライブラリにアドレス参照変換部を設ければ、呼び出す関数の絶対アドレスを意識した開発を行う必要がなく、アプリ共通ライブラリの開発をサービス共通ライブラリと別個に行うことが容易となる。また、アプリ共通ライブラリの開発はアドレス参照変換部を設けることで呼び出す関数の絶対アドレスを開発時に決定する必要がないので、アプリ共通ライブラリのバージョンアップを容易に行える。従って、アプリ共通ライブラリの開発の作業効率を向上させることができる。

【００３２】また、請求項９にかかる発明は、請求項６～８のいずれか一つに記載の画像形成装置において、画像形成装置の起動時に、前記関数配置情報に基づいて、前記呼び出す関数の絶対アドレスを前記アドレス参照変換部に格納するアドレス参照解決手段をさらに備えたこ

とを特徴とする。

【００３３】この請求項９に記載の発明によれば、アドレス参照解決手段によって、画像形成装置の起動時に関数配置情報に基づいて、呼び出す関数の絶対アドレスをアドレス参照変換部に格納するので、実行時のアドレス解決の処理が簡易なものとなり、プログラム実行時のオーバヘッドを短縮することができる。

【００３４】また、請求項１０にかかる発明は、請求項９に記載の画像形成装置において、前記アドレス参照解決手段は、前記サービス共通ライブラリまたは前記アプリ共通ライブラリの先頭の絶対アドレスと該先頭の絶対アドレスからの関数のオフセットを前記アドレス参照変換部に格納することを特徴とする。

【００３５】この請求項１０に記載の発明によれば、アドレス参照解決手段がサービス共通ライブラリまたはアプリ共通ライブラリの先頭の絶対アドレスと該先頭の絶対アドレスからの関数のオフセットをアドレス参照変換部に格納することで、共通ライブラリやアプリ共通ライブラリの先頭の絶対アドレスを定めておけば、関数配置情報には先頭アドレスからのオフセットを格納しておけばよく、サービス共通ライブラリまたはアプリ共通ライブラリの開発の作業効率を向上させることができる。

【００３６】また、請求項１１にかかる発明は、請求項６～１０のいずれか一つに記載の画像形成装置において、前記コントロールサービスおよび前記アプリケーションは、一または複数のスレッドを含むプロセスとして動作し、前記サービス共通ライブラリは、前記スレッドに関する処理を実行する関数を含んでいることを特徴とする。

【００３７】この請求項１１に記載の発明によれば、サービス共通ライブラリがスレッドに関する処理を実行する関数を含んでいることで、スレッドに関する関数を共通化してコントロールサービスのアプリケーションの開発の作業効率を向上させることができる。

【００３８】また、請求項１２にかかる発明は、画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションと、前記アプリケーションと前記ハードウェア資源との間に介在し、ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行う複数のコントロールサービスを含むプラットホームと、複数の前記アプリケーションから共通に呼び出される一または複数の関数と各関数のメモリ上での絶対アドレスを示す関数配置情報とを含むアプリ共通ライブラリと、前記複数のコントロールサービスから共通に呼び出される一または複数の関数と各関数のメモリ上での絶対アドレスを示す関数配置情報とを有するサービス共通ライブラリとを備えた画像形成装置を起動する際に、前記コントロールサービスおよび前記アプリケーションから呼び出される関数のアドレスを解決するアドレス解決方法において、前記呼び出す関数のメモリ上での絶対アドレスを、前記関数配置情報に基づいて、前記アプリケーションおよび前記コントロールサービスのアドレス参照変換部に格納するアドレス参照解決ステップを含んだことを特徴とする。

【００３９】この請求項１２に記載の発明によれば、呼び出す関数のメモリ上での絶対アドレスを、前記関数配置情報に基づいて、前記アプリケーションおよび前記コントロールサービスのアドレス参照変換部に格納するアドレス参照解決ステップを含んだことで、実行時のアドレス解決の処理が簡易なものとなり、プログラム実行時のオーバヘッドを短縮することができる。

【００４０】また、請求項１３にかかる発明は、請求項１２に記載のアドレス解決方法において、前記アドレス参照解決ステップは、前記サービス共通ライブラリまたは前記アプリ共通ライブラリの先頭の絶対アドレスと該先頭の絶対アドレスからの関数のオフセットを前記アドレス参照変換部に格納することを特徴とする。

【００４１】この請求項１３に記載の発明によれば、アドレス参照解決ステップがサービス共通ライブラリまたはアプリ共通ライブラリの先頭の絶対アドレスと該先頭の絶対アドレスからの関数のオフセットを前記アドレス参照変換部に格納することで、共通ライブラリやアプリ共通ライブラリの先頭の絶対アドレスを定めておけば、関数配置情報には先頭アドレスからのオフセットを格納しておけばよく、サービス共通ライブラリまたはアプリ共通ライブラリの開発の作業効率を向上させることができる。

【００４２】

【発明の実施の形態】以下に添付図面を参照して、この発明によるプログラム作成装置、プログラム作成方法、その方法をコンピュータに実行させるプログラム、画像形成装置およびアドレス解決方法の好適な実施の形態を詳細に説明する。

【００４３】（実施の形態１）

（プログラム作成装置のハードウエア構成）まず、この発明の実施の形態によるプログラム作成装置のハードウエア構成について説明する。図１は、この発明の実施の形態によるプログラム作成装置のハードウエア構成を示す説明図である。

【００４４】同図において、１０１は装置全体を制御するＣＰＵを、１０２は基本入出力プログラムを記憶したＲＯＭを、１０３はＣＰＵ１０１のワークエリアとして使用されるＲＡＭを、それぞれ示している。

【００４５】また、１０４はＣＰＵ１０１の制御にしたがってＨＤ（ハードディスク）１０５に対するデータのリード／ライトを制御するＨＤＤ（ハードディスクドライブ）を、１０５はＨＤＤ１０４の制御にしたがって書き込まれたデータを記憶するＨＤを、それぞれ示してい

る。

【００４６】また、１０６はＣＰＵ１０１の制御にしたがってＦＤ（フロッピー（登録商標）ディスク）１０７に対するデータのリード／ライトを制御するＦＤＤ（フロッピーディスクドライブ）を、１０７はＦＤＤ１０６の制御にしたがって書き込まれたデータを記憶する着脱自在のＦＤを、それぞれ示している。

【００４７】また、１０８はカーソル、メニュー、ウィンドウ、あるいは文字や画像などの各種データを表示するディスプレイを、１０９は通信回線１１０を介してネットワークＮＥＴに接続され、そのネットワークとＣＰＵ１０１とのインターフェースとして機能するネットワークボードを、それぞれ示している。また、１１１は文字、数値、各種指示などの入力のための複数のキーを備えたキーボードを、１１２は各種指示の選択や実行、処理対象の選択、カーソルの移動などをおこなうマウスを、それぞれ示している。

【００４８】また、１１３は文字や画像を光学的に読み取るスキャナを、１１４はＣＰＵ１０１の制御にしたがって文字や画像を印刷するプリンタを、１１５は着脱可能な記録媒体であるＣＤ－ＲＯＭを、１１６はＣＤ－ＲＯＭ１１５に対するデータのリードを制御するＣＤ－ＲＯＭドライブを、１００は上記各部を接続するためのバスまたはケーブルを、それぞれ示している。

【００４９】（プログラム作成装置の機能的構成）つぎに、この発明の実施の形態によるプログラム作成装置の機能的構成について説明する。図２は、この発明の実施の形態によるプログラム作成装置の構成を機能的に示す説明図である。この発明の実施の形態によるプログラム作成装置は、プログラム記憶部２００と、アドレス参照変換モジュール作成部２０１と、プログラム変換部２０２と、プログラム結合部２０３と、プログラムロード部２０４とを含む構成である。

【００５０】２００は、プログラム記憶部であり、実行プログラム記憶部２００ａと、アドレス参照変換モジュール記憶部２００ｂとを含む構成である。まず、実行プログラム記憶部２００ａは、実行可能な形式のプログラムを複数のファイルに分類・整理して記憶している。ここに記憶されるプログラムファイルには、大別して

（ａ）他の複数のプログラムファイルから共通に参照されるモジュール群が記述されたファイル、すなわち共有プログラムファイルと、（ｂ）上記モジュールを、後述するアドレス参照変換モジュールを介して参照するプログラムの記述されたファイルとがある。

【００５１】図３は、実行プログラム記憶部２００ａに記憶される共有プログラムファイル３００の一例を模式的に示す説明図である。この共有ファイルには、他のプログラムから呼び出されて利用される外部関数モジュールｆ１、ｆ２、ｆ３およびｆ４があらかじめスタティックリンクされ、それぞれアドレスＡ１、Ａ２、Ａ３およびＡ４に格納されているものとする。また、各共有ファイルには当該ファイルに含まれる外部関数モジュールの名称とアドレスとを対応づけた、シンボルテーブル３００ａが格納されている。

【００５２】また、アドレス参照変換モジュール記憶部（アドレス参照変換モジュールアーカイブ）２００ｂは、後述するアドレス参照変換モジュール作成部２０１により作成されたアドレス参照変換モジュールを記憶している。このアドレス参照変換モジュールについてはすぐ後に説明する。

【００５３】つぎに２０１は、アドレス参照変換モジュール作成部であり、実行プログラム記憶部２００ａに記憶された共有プログラムファイル内の、それぞれの外部関数モジュールに対応するアドレス参照変換モジュールを作成する。そして、作成したアドレス参照変換モジュールを上述のアドレス参照変換モジュール記憶部２００ｂに格納する。

【００５４】図４は、アドレス参照変換モジュール作成部２０１により作成され、アドレス参照変換モジュール記憶部２００ｂに格納されたアドレス参照変換モジュールの一例を模式的に示す説明図である。たとえば、４００は図３に示す外部関数ｆ１に対応するアドレス参照変換モジュールであり、外部関数ｆ１と同名のアドレス参照変換ｆ１’と、外部関数ｆ１の実行時のアドレス値が格納されるデータ領域ｄ１とが定義されている。この関数ｆ１’は、「データ領域ｄ１に格納されているアドレスにジャンプする」という処理のみをおこなう関数である。

【００５５】また、データ領域ｄ１にはこの時点では０、すなわちありえないアドレス値が格納されており、後述のようにプログラムの実行時に、プログラムロード部２０４により外部関数ｆ１の現実のアドレス値が設定される。なお、外部関数ｆ２～ｆ４に対応するアドレス参照変換モジュール４０１～４０３も形式は同一であり、関数名称さえわかれば機械的に作成することができる。

【００５６】２０２は、プログラム変換部（コンパイラ）であり、キーボード１１１から入力されたりＨＤ１０５から読み出されたりしたソースプログラムを、コンピュータで実行可能な形式のオブジェクトプログラムに変換する。

【００５７】２０３は、プログラム結合部（スタティックリンカ）であり、上記プログラムが参照している外部関数に対応するアドレス参照変換モジュール、すなわち当該外部関数と同名の関数を有するアドレス参照変換モジュールを、アドレス参照変換モジュール記憶部２００ｂから検索して上記プログラムに結合する。そして、必要な関数が結合された実行可能なプログラムファイルを、上述の実行プログラム記憶部２００ａに格納する。

【００５８】図５は、プログラム変換部２０２により機

械語に変換され、プログラム結合部２０３により必要な関数が結合された後、実行プログラム記憶部２００ａに格納されたプログラムの一例を模式的に示す説明図である。このプログラムがもともと外部関数ｆ１、ｆ２およびｆ３を参照していたものとすると、変換・結合後のプログラムにはそれらの外部関数の代わりに、対応するアドレス参照変換モジュールｆ１'、ｆ２'およびｆ３'が結合されている。ｄ１、ｄ２およびｄ３の各データ領域の値は、まだ設定されないままである。

【００５９】つぎに２０４は、プログラムロード部（ランタイムローダ）であり、アドレス検索部２０４ａとアドレス設定部２０４ｂとを含む構成である。このプログラムロード部２０４は、実行プログラム記憶部２００ａに記憶されたプログラムの起動処理をおこなう。ここでは、図５に示すプログラムを起動する場合を例として説明する。

【００６０】まず、プログラムロード部２０４は、上記プログラムが参照しているプログラムファイル、この例では図３に示す共有プログラムファイル３００を実行プログラム記憶部２００ａから検索する。そして、この共有プログラムファイル３００のシンボルテーブル３００ａをアドレス検索部２０４ａにより検索し、上記プログラムに結合されているアドレス参照変換と同名の関数、すなわち当該アドレス参照変換に対応する外部関数のアドレス値を取得する。この例では、アドレス参照変換ｆ１'、ｆ２'およびｆ３'に対応する、外部関数ｆ１、ｆ２およびｆ３のアドレス値Ａ１、Ａ２およびＡ３が取得される。

【００６１】つぎに、プログラムロード部２０４はアドレス設定部２０４ｂにより、上記アドレス値を上記プログラムのデータ領域ｄ１、ｄ２およびｄ３にそれぞれ格納する。これにより、アドレス参照変換ｆ１'、ｆ２'およびｆ３'によるジャンプ先が確定され、上記プログラムを実行する準備が整う。このプログラムの実行時には、結合されたアドレス参照変換モジュールを経由して、対応する外部関数が実行されることになる。

【００６２】なお、アドレス参照変換モジュール作成部２０１、プログラム変換部２０２、プログラム結合部２０３およびプログラムロード部２０４は、それぞれＨＤ１０５やＦＤ１０７、あるいはＣＤ－ＲＯＭ１１５などの各種記録媒体からＲＡＭ１０３に読み出されたプログラムの命令にしたがって、ＣＰＵ１０１などが命令処理を実行することにより、各部の機能を実現するものである。

【００６３】（プログラム作成から実行までの処理の流れ）つぎに、この発明の実施の形態によるプログラム作成装置における、プログラムの作成から実行までの処理の流れを説明する。図６は、この発明の実施の形態によるプログラム作成装置の、プログラム作成～実行処理の流れを示すフローチャートである。

【００６４】なお、下記（１）～（３）の処理は必ずしも連続しておこなわれるものでなく、各処理の間に時間的な間隔があくこともあるが、同図では分かりやすく全体の流れを示した。

【００６５】（１）前準備

ステップＳ６０１～Ｓ６０３で、まずアドレス参照変換モジュール作成部２０１において、それぞれの外部関数モジュールに対応するアドレス参照変換モジュールを準備しておく。ステップＳ６０１で、アドレス参照変換モジュール作成部２０１は実行プログラム記憶部２００ａに保持されている共有プログラムファイル３００のシンボルテーブルを検索して、それらのファイルに記述されている外部関数の名称を順次取得する。

【００６６】続くステップＳ６０２で、各外部関数へのジャンプ命令からなる、当該外部関数と同一名称の関数と、当該関数のジャンプ先のアドレスが格納されるデータ領域とからなるアドレス参照変換モジュールを作成する。そして、作成したアドレス参照変換モジュールを、ステップＳ６０３でアドレス参照変換モジュール記憶部２００ｂに格納しておく。

【００６７】（２）変換と結合

この後この装置に、変換・結合が必要なプログラムが入力すると、ステップＳ６０４でプログラム変換部２０２は入力したソースコードをオブジェクトコードに変換する（コンパイル）。そして、ステップＳ６０５でプログラム結合部２０３は、上記プログラムで参照されている外部関数と同名の関数をアドレス参照変換モジュール記憶部２００ｂから検索し、検索された関数を含むアドレス参照変換モジュールを、ステップＳ６０６で上記プログラムに結合する（リンク）。そして、機械語に変換され、必要なアドレス参照変換モジュールが結合された上記プログラムを、ステップＳ６０７で実行プログラム記憶部２００ａに格納しておく。

【００６８】（３）実行

そして、この後この装置に上記プログラムの実行指示が入力すると、ステップＳ６０８でプログラムロード部２０４は、実行プログラム記憶部２００ａから指示されたプログラムを読み込む。そして、ステップＳ６０９でまず当該プログラムが参照している共有プログラムファイル３００を、実行プログラム記憶部２００ａから読み込む。

【００６９】さらに、ステップＳ６１０でプログラムロード部２０４のアドレス検索部２０４ａは、ステップＳ６０９で検索されたファイルのシンボルテーブルを検索して、上記プログラムに結合されているアドレス参照変換と同名の外部関数のアドレスを順次取得する。そして、このアドレスをステップＳ６１１で、アドレス設定部２０４ｂにより上記プログラムのアドレス参照変換モジュールに設定する。そして、このアドレスの確定により完全に実行可能となった上記プログラムを、ステップ

Ｓ６１２で実行する。

【００７０】以上説明した実施の形態によれば、外部関数の実体を含む共有プログラムファイルも、アドレス参照変換モジュールを経由して当該外部関数を参照するプログラムも、完全にスタティックリンクされているため、オブジェクトを再配置可能にするための命令コードが一切含まれず、プログラムのサイズがダイナミックリンク方式の場合よりも小さくなる。

【００７１】また、従来のスタティックリンク方式とは異なり、外部関数の実体を含むプログラムファイルを複数のプログラムから参照することができるため、複数のプログラムで共通に利用する外部関数群を共有ライブラリとしてまとめることにより、プログラム全体のサイズを削減することができる。

【００７２】さらに、外部関数を参照するプログラムには当該関数の実体が含まれていないので、不具合などの理由により外部関数の内部が修正された場合でも、共有プログラムファイルだけを更新すればよく、プログラムの開発効率が向上する。

【００７３】さらに、本発明のスタティックリンク方式は、ソフトウエア開発ツールであるコンパイラやスタティックリンカに一切変更を加える必要なく実現できるため、従来の開発環境と共存させることが可能であり、ソフトウエア開発の効率を落とすことなく、上述したメリットを享受できる。

【００７４】（実施の形態２）図７は、この発明の実施の形態２である画像形成装置（以下、「複合機」という）の構成を示すブロック図である。図７に示すように、複合機７００は、白黒ラインプリンタ（B&W LP）７０１と、カラーラインプリンタ（Color LP）７０２と、スキャナ、ファクシミリなどのハードウェアリソース７０３などを有するとともに、プラットホーム７２０とアプリケーション７３０とから構成されるソフトウェア群７１０とを備えている。

【００７５】プラットホーム７２０は、アプリケーションからの処理要求を解釈してハードウェア資源の獲得要求を発生させるコントロールサービスと、一または複数のハードウェア資源の管理を行い、コントロールサービスからの獲得要求を調停するシステムリソースマネージャ（ＳＲＭ）７２３と、汎用ＯＳ７２１とを有する。

【００７６】コントロールサービスは、複数のサービスモジュールから形成され、ＳＣＳ（システムコントロールサービス）７２２と、ＥＣＳ（エンジンコントロールサービス）７２４と、ＭＣＳ（メモリコントロールサービス）７２５と、ＯＣＳ（オペレーションパネルコントロールサービス）７２６と、ＦＣＳ（ファックスコントロールサービス）７２７と、ＮＣＳ（ネットワークコントロールサービス）７２８とから構成される。なお、このプラットホーム７２０は、あらかじめ定義された関数により前記アプリケーション７３０から処理要求を受信可能とするアプリケーションプログラムインタフェース（ＡＰＩ）を有する。

【００７７】汎用ＯＳ７２１は、ＵＮＩＸ（登録商標）などの汎用オペレーティングシステムであり、プラットホーム７２０並びにアプリケーション７３０の各ソフトウェアをそれぞれプロセスとして並列実行する。

【００７８】コントロールサービスのそれぞれは、プロセスとして動作し、プロセス内に一または複数のスレッドを並列実行している。

【００７９】ＳＲＭ７２３のプロセスは、ＳＣＳ７２２とともにシステムの制御およびリソースの管理を行うものであり、スキャナ部やプリンタ部などのエンジン、メモリ、ＨＤＤファイル、ホストＩ／Ｏ（セントロＩ／Ｆ、ネットワークＩ／Ｆ、ＩＥＥＥ１３９４　Ｉ／Ｆ、ＲＳ２３２Ｃ　Ｉ／Ｆなど）のハードウェア資源を利用する上位層からの要求にしたがって調停を行い、実行制御を行う。

【００８０】具体的には、このＳＲＭ７２３は、要求されたハードウェア資源が利用可能であるか（他の要求により利用されていないかどうか）を判断し、利用可能であれば要求されたハードウェア資源が利用可能である旨を上位層に伝える。また、ＳＲＭ７２３は、上位層からの要求に対してハードウェア資源の利用スケジューリングを行い、要求内容（例えば、プリンタエンジンにより紙搬送と作像動作、メモリ確保、ファイル生成など）を直接実施している。

【００８１】ＳＣＳ７２２のプロセスは、アプリ管理、操作部制御、システム画面表示、ＬＥＤ表示、リソース管理、割り込みアプリ制御を行う。

【００８２】ＥＣＳ７２４のプロセスは、白黒ラインプリンタ（B&W LP）７０１、カラーラインプリンタ（Color LP）７０２、スキャナ、ファクシミリなどからなるハードウェアリソース７０３のエンジンの制御を行う。

【００８３】ＭＣＳ７２５のプロセスは、画像メモリの取得および解放、ハードディスク装置（ＨＤＤ）の利用、画像データの圧縮および伸張などを行う。

【００８４】ＯＣＳ７２６のプロセスは、オペレータと本体制御間の情報伝達手段となる操作パネル（オペレーションパネル）の制御を行う。

【００８５】ＦＣＳ７２７のプロセスは、システムコントローラの各アプリ層からＰＳＴＮ／ＩＳＤＮ網を利用したファクシミリ送受信、ＢＫＭ（バックアップＳＲＡＭ）で管理されている各種ファクシミリデータの登録／引用、ファクシミリ読みとり、ファクシミリ受信印刷、融合送受信を行うためのＡＰＩを提供する。

【００８６】ＮＣＳ７２８は、ネットワークＩ／Ｏを必要とするアプリケーションに対して共通に利用できるサービスを提供するためのプロセスであり、ネットワーク側から各プロトコルによって受信したデータを各アプリケーションに振り分けたり、アプリケーションからデー

タをネットワーク側に送信する際の仲介を行う。

【００８７】アプリケーション７３０は、プリンタ、コピー、スキャナまたはファクシミリなどの画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うものである。アプリケーション７３０は、ページ記述言語（ＰＤＬ）、ＰＣＬおよびポストスクリプト（ＰＳ）を有するプリンタ用のアプリケーションであるプリンタアプリ７１１と、コピー用アプリケーションであるコピーアプリ７１２と、ファクシミリ用アプリケーションであるファックスアプリ７１３と、スキャナ用アプリケーションであるスキャナアプリ７１４と、ネットワークファイル用アプリケーションであるネットファイルアプリ７１５と、工程検査用アプリケーションである工程検査アプリ７１６とを有している。

【００８８】図８は、実施の形態２にかかる複合機におけるコントロールサービスとサービス共通ライブラリとランタイムローダの関係を示す模式図である。サービス共通ライブラリ８４０は、ＥＣＳ７２４、ＭＣＳ７２５、ＳＣＳ７２２などの複数のコントロールサービスから共通して呼び出される複数の関数を一つのファイルにまとめ、複数のコントロールサービスから利用できるようにしたものである。このサービス共通ライブラリ８４０には、プロセスの生成、オープンおよびクローズなどのプロセスに関する処理を行う関数、スレッドの生成、オープンおよびクローズなどのスレッドに関する処理を行う関数、ファイルのオープンやクローズなどのファイル操作の関数などが登録されている。また、サービス共通ライブラリ８４０には、登録された関数の絶対アドレスを示す関数配置情報８４４が格納されている。

【００８９】コントロールサービスは、実行時に呼び出す関数の絶対アドレスが格納される領域と、当該領域に格納された絶対アドレスにジャンプする命令が記述されたアドレス参照変換部を有している。

【００９０】ランタイムローダ８４３は、アプリケーション７３０およびシステムコントロールサービスの実行時に、サービス共通ライブラリ８４０の関数配置情報８４４から各関数の絶対アドレスを取得して、各コントロールサービスのアドレス参照変換部８４２に取得した関数の絶対アドレスを格納することにより、アドレス解決を行うものである。このランタイムローダ８４３は、本発明におけるアドレス参照解決手段を構成する。

【００９１】図１０は、実施の形態２にかかる複合機におけるサービス共通ライブラリ８４０の構造を示す模式図である。このサービス共通ライブラリ８４０には、各関数の実体部分と、各関数の仮想メモリ空間上での絶対アドレスを示す関数配置情報８４４とを有している。図１０では、関数の一例として、ｔｈｒｅａｄ＿ｃｒｅａｔｅ関数と、ｆｉｌｅ＿ｏｐｅｎ関数がサービス共通ライブラリ８４０に登録されている。

【００９２】サービス共通ライブラリ８４０の関数配置情報８４４には、ｔｈｒｅａｄ＿ｃｒｅａｔｅ関数の絶対アドレスであるａｄｄｒ１と、ｆｉｌｅ＿ｏｐｅｎ関数の絶対アドレスであるａｄｄｒ２が格納されている。なお、サービス共通ライブラリ８４０の先頭の絶対アドレスをａｄｄｒ０、ｔｈｒｅａｄ＿ｃｒｅａｔｅ関数のサービス共通ライブラリ８４０先頭からのオフセットをｏｆｆｓｅｔ１、ｆｉｌｅ＿ｏｐｅｎ関数のサービス共通ライブラリ８４０先頭からのオフセットをｏｆｆｓｅｔ２として、ｔｈｒｅａｄ＿ｃｒｅａｔｅ関数の配置情報をａｄｄｒ０＋ｏｆｆｓｅｔ１、ｆｉｌｅ＿ｏｐｅｎ関数の配置情報をａｄｄｒ０＋ｏｆｆｓｅｔ２のように格納しても良い。

【００９３】アプリ共通ライブラリ８４１は、プリンタアプリ７１１、コピーアプリ７１０などの複数のアプリケーションから共通して呼び出される複数の関数を一つのファイルにまとめ、複数のアプリケーションから利用できるようにしたものである。このアプリ共通ライブラリ８４１にも、プロセスに関する処理を行う関数、スレッドに関する処理を行う関数、ファイル操作の関数などが登録されている。アプリ共通ライブラリ８４１も、各関数の実体と関数配置情報８４４とを有しており、その構造についてはサービス共通ライブラリ８４０と同様である。

【００９４】つぎに、このように構成された実施の形態２にかかる複合機におけるコントロールサービス実行時の動作について、ＳＣＳ７２２を例にあげて説明する。図９は、ＳＣＳ７２２のプロセス起動時における仮想メモリ上の配置を示す模式図である。図９に示すように。ＳＣＳ７２２の本体プログラムがアドレスの下位に配置され、サービス共通ライブラリ８４０がアドレスの上位に配置される。

【００９５】図１１は、ＳＣＳ７２２の本体プログラムの内容を示す説明図である。なお、図１１では、サービス共通ライブラリ８４０の関数呼び出しを中心に示している。

【００９６】図１１に示すように、ＳＣＳ７２２の本体プログラムでは、サービス共通ライブラリ８４０のｔｈｒｅａｄ＿ｃｒｅａｔｅ関数を呼び出している。そして、ＳＣＳ７２２の本体プログラムは、当該関数呼び出しのためのアドレス参照変換部８４２を有している。このアドレス参照変換部８４２には、ｔｈｒｅａｄ＿ｃｒｅａｔｅ関数の絶対アドレスが格納される領域ｄ１が確保され、ｄ１に格納される絶対アドレスの関数を呼び出す旨の処理が記述されている。

【００９７】図１２は、実施の形態２の複合機におけるランタイムローダ８４３のアドレス解決処理の手順を示すフローチャートである。ＳＣＳ７２２が起動されると、ランタイムローダ８４３がＳＣＳ７２２の本体プログラムの中から、外部関数として定義されている関数シンボルを検索する（ステップＳ１２０１）。これによ

り、ｔｈｒｅａｄ＿ｃｒｅａｔｅ関数が検索されそのシンボルｔｈｒｅａｄ＿ｃｒｅａｔｅが検索結果として得られる。なお、この時点では、ＳＣＳ７２２の本体プログラムのアドレス参照変換部８４２の領域ｄ１には現実には存在しないアドレス（たとえば、０ｘ００００００００など）が格納されており、まだアドレス解決がなされていない状態となっている。

【００９８】そして、外部関数の関数シンボルの有無を判断し（ステップＳ１２０２）、関数シンボルがある場合には、サービス共通ライブラリ８４０の関数配置情報８４４を参照し、検索された関数シンボルの関数の絶対アドレスを関数配置情報８４４から取得する（ステップＳ１２０３）。

【００９９】つぎに、取得した絶対アドレスを本体プログラムのアドレス参照変換部８４２の該当する関数の領域ｄ１に格納する（ステップＳ１２０４）。これにより、絶対アドレスａｄｄｒ１が領域ｄ１に格納され、ｔｈｒｅａｄ＿ｃｒｅａｔｅ関数のアドレス解決がなされる。

【０１００】ステップＳ１２０１で検索されたすべての関数シンボルに対して、上述のステップＳ１２０３およびＳ１２０４の処理を繰り返すことにより（ステップＳ１２０５）、ＳＣＳ７２２の本体プログラムの外部関数のアドレス解決処理が完了する。

【０１０１】そして、すべての関数のアドレス解決が完了すると、ＳＣＳ７２２の本体プログラムのｍａｉｎの実行が開始される。ｍａｉｎの実行中に、スレッドを生成するためにｔｈｒｅａｄ＿ｃｒｅａｔｅ関数が呼び出されると、アドレス参照変換部８４２によってｔｈｒｅａｄ＿ｃｒｅａｔｅ関数の絶対アドレスａｄｄｒ１に制御が移り、サービス共通ライブラリ８４０のｔｈｒｅａｄ＿ｃｒｅａｔｅ関数の実体が呼び出されてスレッドが生成される。

【０１０２】なお、実施の形態２では、ＳＣＳ７２２の実行について説明したが、他のコントロールサービスやアプリケーションの実行についても同様の処理が行われる。ただし、アプリケーションの実行の際には、アプリ共通ライブラリ８４１の中の関数との間でアドレス解決処理がなされることになる。

【０１０３】このように、実施の形態２にかかる複合機では、複数のコントロールサービスから共通に呼び出される関数の実体と、各関数の仮想メモリ上での絶対アドレスを示す関数配置情報８４４とを有するサービス共通ライブラリ８４０を備え、各コントロールサービスは、ランタイムローダ８４３により関数の絶対アドレスが実行時に格納されるアドレス参照変換部８４２を有しているので、関数の絶対アドレスを意識した開発を行う必要がなく、コントロールサービスの開発を共通ライブラリや他のコントロールサービスと別個に行うことが容易に行える。また、コントロールサービスの開発はアドレス参照変換部８４２を設けることで呼び出す関数の絶対アドレスを開発時に決定する必要がないので、コントロールサービスのバージョンアップを容易に行え、コントロールサービスの開発の作業効率を向上させることができる。

【０１０４】また、実施の形態２にかかる複合機では、ランタイムローダ８４３によって、コントロールサービスの起動時に関数配置情報８４４に基づいて関数の絶対アドレスをアドレス参照変換部８４２に格納するので、実行時のアドレス解決の処理が簡易なものとなり、プログラム実行時のオーバヘッドが短縮される。

【０１０５】（実施の形態３）実施の形態２の複合機では、複数のコントロールサービスから呼び出される関数を登録したサービス共通ライブラリ８４０と、複数のアプリケーションから呼び出される関数を登録したアプリ共通ライブラリ８４１とを設け、各共通ライブラリは独立したものであったが、この実施の形態３にかかる複合機は、アプリ共通ライブラリ８４１に登録された関数からさらにサービス共通ライブラリ８４０に登録されている関数を呼び出しているものである。

【０１０６】実施の形態３にかかる複合機の構成は、実施の形態２の複合機と同様であるので説明を省略する。図１４は、実施の形態３にかかる複合機におけるアプリ共通ライブラリ８４１の構造を示す模式図である。このアプリ共通ライブラリ８４１には、各関数の実体部分と、各関数の仮想メモリ空間上での絶対アドレスを示す関数配置情報８４４とを有している。図１４では、関数の一例として、ｆｏｎｔ＿ｓｅａｒｃｈ関数がアプリ共通ライブラリ８４１に登録されており、ｆｏｎｔ＿ｓｅａｒｃｈ関数の絶対アドレスａｄｄｒ３を示す関数配置情報８４４を有している。さらに、図１４に示すように、アプリ共通ライブラリ８４１のｆｏｎｔ＿ｓｅａｒｃｈ関数の中から、サービス共通ライブラリ８４０のｆｉｌｅ＿ｏｐｅｎ関数を呼び出すようになっている。このため、アプリ共通ライブラリ８４１には、ｆｉｌｅ＿ｏｐｅｎ関数呼び出しのためのアドレス参照変換部８４２を有している。このアドレス参照変換部８４２には、ｆｉｌｅ＿ｏｐｅｎ関数の絶対アドレスが格納される領域ｄ２が確保され、ｄ２に格納される絶対アドレスの関数を呼び出す旨の処理が記述されている。

【０１０７】つぎに、このように構成された実施の形態３にかかる複合機におけるアプリケーション実行時の動作について、プリンタアプリ７１１を例にあげて説明する。図１３は、プリンタアプリ７１１のプロセス起動時における仮想メモリ上の配置を示す模式図である。図１３に示すように、プリンタアプリ７１１の本体プログラムがアドレスの下位に配置され、その上位側にアプリ共通ライブラリ８４１が、さらにその上位側にサービス共通ライブラリ８４０が配置される。なお、サービス共通ライブラリ８４０が配置されるアドレスは、実施の形態

２におけるサービス共通ライブラリ８４０と同一アドレスに固定されている。すなわち、各ライブラリの配置は絶対アドレスで固定された位置に配置されるようになっている。

【０１０８】図１５は、プリンタアプリ７１１の本体プログラムの内容を示す説明図である。なお、図１５では、アプリ共通ライブラリ８４１およびービス共通ライブラリの関数呼び出しを中心に示している。

【０１０９】図１５に示すように、プリンタ７１１の本体プログラムでは、アプリ共通ライブラリ８４１のｆｏｎｔ＿ｓｅａｒｃｈ関数とサービス共通ライブラリ８４０のｔｈｒｅａｄ＿ｃｒｅａｔｅ関数を呼び出している。そして、プリンタアプリ７１１の本体プログラムは、これらの関数呼び出しのためのアドレス参照変換部８４２を有している。このアドレス参照変換部８４２には、ｆｏｎｔ＿ｓｅａｒｃｈ関数の絶対アドレスが格納される領域ｄ３が確保され、領域ｄ３に格納される絶対アドレスの関数を呼び出す旨の処理が記述されている。また、このアドレス参照変換部８４２には、ｔｈｒｅａｄ＿ｃｒｅａｔｅ関数の絶対アドレスが格納される領域ｄ４が確保され、ｄ４に格納される絶対アドレスの関数を呼び出す旨の処理が記述されている。なお、プリンタアプリ７１１の起動前は、領域ｄ３、ｄ４には、現実に存在しないアドレス（たとえば、０ｘ００００００００など）が格納され、アドレスは解決されていない状態となっている。

【０１１０】プリンタアプリ７１１の本体プログラムが起動されると、ランタイムローダ８４３はの関数のアドレス解決処理が行われる。すなわち、アプリ共通ライブラリ８４１から呼び出される外部関数の関数シンボルを検索する。そして、検索された関数シンボルｆｉｌｅ＿ｏｐｅｎの関数の絶対アドレスを、サービス共通ライブラリ８４０の関数配置情報８４４を参照して取得する。これにより絶対アドレスａｄｄｒ２が取得され、この絶対アドレスａｄｄｒ２をアプリ共通ライブラリ８４１内のアドレス参照変換部８４２の領域ｄ２に格納する。

【０１１１】つぎに、ランタイムローダ８４３は、プリンタアプリ７１１の本体プログラム内のアドレス解決処理を行う。すなわち、プリンタアプリ７１１の本体プログラムからから呼び出される外部関数の関数シンボルを検索する。このとき、関数シンボルとして、ｆｏｎｔ＿ｓｅａｒｃｈとｔｈｒｅａｄ＿ｃｒｅａｔｅが検索される。

【０１１２】そして、ランタイムローダ８４３は、まず検索された関数シンボルｆｏｎｔ＿ｓｅａｒｃｈの関数の絶対アドレスを、アプリ共通ライブラリ８４１の関数配置情報８４４を参照して取得する。これにより絶対アドレスａｄｄｒ３が取得され、この絶対アドレスａｄｄｒ３をプリンタアプリ７１１の本体プログラム内のアドレス参照変換部８４２の領域ｄ３に格納する。

【０１１３】つぎに、ランタイムローダ８４３は、もう一つの検索された関数シンボルｔｈｒｅａｄ＿ｃｒｅａｔｅの関数の絶対アドレスを得るために、アプリ共通ライブラリ８４１の関数配置情報８４４を参照する。アプリ共通ライブラリ８４１の関数配置情報８４４には、ｔｈｒｅａｄ＿ｃｒｅａｔｅは登録されていないため、ついでサービス共通ライブラリ８４０の関数配置情報８４４が参照される。そして、サービス共通ライブラリ８４０の関数配置情報８４４からｔｈｒｅａｄ＿ｃｒｅａｔｅの絶対アドレスａｄｄｒ１を取得し、プリンタアプリ７１１の本体プログラムのアドレス参照変換部８４２の領域ｄ４に、絶対アドレスａｄｄｒ１を格納する。これにより、すべてのアドレス解決処理が完了する。

【０１１４】すべてのアドレス解決が完了すると、プリンタアプリ７１１の本体プログラムのｍａｉｎの実行が開始される。ｍａｉｎ実行中に、フォントを検索するためのｆｏｎｔ＿ｓｅａｒｃｈ関数が呼び出されると、アドレス参照変換部８４２によってｆｏｎｔ＿ｓｅａｒｃｈ関数の絶対アドレスａｄｄｒ３に制御が移り、アプリ共通ライブラリ８４１のｆｏｎｔ＿ｓｅａｒｃｈ関数の実体が呼び出され、実行される。

【０１１５】そして、ｆｏｎｔ＿ｓｅａｒｃｈ関数の実行中に、ファイルオープンのためのｆｉｌｅ＿ｏｐｅｎ関数が呼び出されると、アドレス参照変換部８４２によってｆｉｌｅ＿ｏｐｅｎ関数の絶対アドレスａｄｄｒ２に制御が移り、サービス共通ライブラリ８４０のｆｉｌｅ＿ｏｐｅｎ関数の実体が呼び出され、実行される。

【０１１６】つぎに、プリンタアプリ７１１の本体プログラムがスレッドを生成するためにｔｈｒｅａｄ＿ｃｒｅａｔｅ関数が呼び出されると、アドレス参照変換部８４２によってｔｈｒｅａｄ＿ｃｒｅａｔｅ関数の絶対アドレスａｄｄｒ１に制御が移り、サービス共通ライブラリ８４０のｔｈｒｅａｄ＿ｃｒｅａｔｅ関数の実体が呼び出されて実行される。

【０１１７】なお、実施の形態３では、プリンタアプリ７１１の実行について説明したが、他のアプリケーションの実行についても同様の処理が行われる。

【０１１８】このように、実施の形態３にかかる複合機では、サービス共通ライブラリ８４０がアプリ共通ライブラリ８４１から呼び出される関数と各関数のメモリ上での絶対アドレスを示す関数配置情報８４４とを有しており、アプリ共通ライブラリ８４１は呼び出す関数の絶対アドレスが実行時に格納されるアドレス参照変換部８４２を有しているので、呼び出す関数の絶対アドレスを意識した開発を行う必要がなく、アプリ共通ライブラリ８４１の開発をサービス共通ライブラリ８４０と別個に行うことが容易になる。また、アプリ共通ライブラリ８４１の開発はアドレス参照変換部８４２を設けることで呼び出す関数の絶対アドレスを開発時に決定する必要がないので、アプリ共通ライブラリ８４１のバージョンア

ップを容易に行え、アプリ共通ライブラリ８４１の開発の作業効率を向上させることが可能となる。

【０１１９】

【発明の効果】以上説明したように請求項１に記載の発明にかかるプログラム作成装置は、コンピュータに実行させるプログラムを作成するプログラム作成装置において、前記プログラムが参照する外部関数モジュールへのジャンプ命令からなるアドレス参照変換モジュールを作成する作成手段と、前記プログラムをコンピュータで実行可能な形式に変換する変換手段と、前記変換手段により変換されたプログラムに、前記作成手段により作成された、当該プログラムが参照する外部関数モジュールへのジャンプ命令からなるアドレス参照変換モジュールを結合する結合手段と、を備えたので、プログラムから参照される外部関数モジュールは、当該モジュールに代えて上記プログラムにスタティックリンクされた、当該モジュールへのジャンプ命令からなるアドレス参照変換モジュールを経由して実行されることになり、これによって、組み込み型の実行プログラムのサイズを抑え、かつ各モジュールの更新・修正などにも柔軟に対応することが可能なプログラム作成装置が得られるという効果を奏する。

【０１２０】また、請求項２に記載の発明にかかるプログラム作成装置は、前記請求項１に記載の発明において、さらに、前記プログラムの実行時に前記結合手段により結合されたアドレス参照変換モジュールのジャンプ先の外部関数モジュールのアドレスを検索する検索手段と、前記検索手段により検索されたアドレスを前記結合手段により結合されたアドレス参照変換モジュールに設定する設定手段と、を備えたので、アドレス参照変換モジュールのジャンプ先のアドレスは、当該モジュールがスタティックリンクされたプログラムの実行時に確定され、これによって、組み込み型の実行プログラムについて、当該プログラムに組み込まれる各モジュールの更新・修正などにも柔軟に対応することが可能なプログラム作成装置が得られるという効果を奏する。

【０１２１】また、請求項３に記載の発明にかかるプログラム作成方法は、コンピュータに実行させるプログラムを作成するプログラム作成方法において、前記プログラムが参照する外部関数モジュールへのジャンプ命令からなるアドレス参照変換モジュールを作成する作成工程と、前記プログラムをコンピュータで実行可能な形式に変換する変換工程と、前記変換工程で変換されたプログラムに、前記作成工程で作成された、当該プログラムが参照する外部関数モジュールへのジャンプ命令からなるアドレス参照変換モジュールを結合する結合工程と、を含んだので、プログラムから参照される外部関数モジュールは、当該モジュールに代えて上記プログラムにスタティックリンクされた、当該モジュールへのジャンプ命令からなるアドレス参照変換モジュールを経由して実行されることになり、これによって、組み込み型の実行プログラムのサイズを抑え、かつ各モジュールの更新・修正などにも柔軟に対応することが可能なプログラム作成方法が得られるという効果を奏する。

【０１２２】また、請求項４に記載の発明にかかるプログラム作成方法は、前記請求項３に記載の発明において、さらに、前記結合工程で結合されたアドレス参照変換モジュールのジャンプ先の外部関数モジュールのアドレスを検索する検索工程と、前記検索工程で検索されたアドレスを前記結合工程で結合されたアドレス参照変換モジュールに設定する設定工程と、を含んだので、アドレス参照変換モジュールのジャンプ先のアドレスは、当該モジュールがスタティックリンクされたプログラムの実行時に確定され、これによって、組み込み型の実行プログラムについて、当該プログラムに組み込まれる各モジュールの更新・修正などにも柔軟に対応することが可能なプログラム作成方法が得られるという効果を奏する。

【０１２３】また、請求項５に記載の発明は、前記請求項３または請求項４に記載された方法をコンピュータに実行させるプログラムであるので、請求項３または請求項４のいずれか一つの動作をコンピュータによって実行することができるという効果を奏する。

【０１２４】また、請求項６にかかる画像形成装置は、複数のアプリケーションから共通に呼び出される一または複数の関数と、各関数のメモリ上での絶対アドレスを示す関数配置情報を含むアプリ共通ライブラリを備えており、アプリケーションでは呼び出す関数の絶対アドレスが実行時に格納されるアドレス参照変換部を有しているので、アプリケーションにアドレス参照変換部を設ければ、呼び出す関数の絶対アドレスを意識した開発を行う必要がなくアプリケーションの開発をアプリ共通ライブラリや他のアプリケーションと別個に行うことが容易に行え、また、アプリケーションの開発はアドレス参照変換部を設けることで呼び出す関数の絶対アドレスを開発時に決定する必要がないので、アプリケーションのバージョンアップを容易に行え、これによって、アプリケーションの開発の作業効率を向上させることができるという効果を奏する。

【０１２５】また、請求項７にかかる画像形成装置は、前記請求項６に記載の発明において、複数のコントロールサービスから共通に呼び出される一または複数の関数と、各関数のメモリ上での絶対アドレスを示す関数配置情報とを有するサービス共通ライブラリをさらに備え、前記コントロールサービスは、呼び出す関数の絶対アドレスが実行時に格納されるアドレス参照変換部を有することで、コントロールサービスにアドレス参照変換部を設ければ、呼び出す関数の絶対アドレスを意識した開発を行う必要がなく、コントロールサービスの開発をサービス共通ライブラリや他のコントロールサービスと別個

に行うことが容易となり、コントロールサービスの開発はアドレス参照変換部を設けることで呼び出す関数の絶対アドレスを開発時に決定する必要がないので、コントロールサービスのバージョンアップを容易に行えるので、コントロールサービスの開発の作業効率を向上させることができるという効果を奏する。

【０１２６】また、請求項８にかかる画像形成装置は、請求項７に記載の発明において、サービス共通ライブラリはさらにアプリ共通ライブラリから呼び出される一または複数の関数と各関数のメモリ上での絶対アドレスを示す関数配置情報とを有し、アプリ共通ライブラリは呼び出す関数の絶対アドレスが実行時に格納されるアドレス参照変換部を有することで、アプリ共通ライブラリにアドレス参照変換部を設ければ、呼び出す関数の絶対アドレスを意識した開発を行う必要がなく、アプリ共通ライブラリの開発をサービス共通ライブラリと別個に行うことが容易となり、また、アプリ共通ライブラリの開発はアドレス参照変換部を設けることで呼び出す関数の絶対アドレスを開発時に決定する必要がないので、アプリ共通ライブラリのバージョンアップを容易に行え、これにより、アプリ共通ライブラリの開発の作業効率を向上させることができるという効果を奏する。

【０１２７】また、請求項９にかかる画像形成装置は、請求項６～８のいずれか一つに記載の発明において、画像形成装置の起動時に関数配置情報に基づいて、呼び出す関数の絶対アドレスをアドレス参照変換部に格納するアドレス参照解決手段を備えたことで、実行時のアドレス解決の処理が簡易なものとなり、プログラム実行時のオーバヘッドを短縮することができるという効果を奏する。

【０１２８】また、請求項１０にかかる画像形成装置は、請求項９に記載の発明において、アドレス参照解決手段がサービス共通ライブラリまたはアプリ共通ライブラリの先頭の絶対アドレスと該先頭の絶対アドレスからの関数のオフセットをアドレス参照変換部に格納することで、サービス共通ライブラリやアプリ共通ライブラリの先頭の絶対アドレスを定めておけば、関数配置情報には先頭アドレスからのオフセットを格納しておけばよく、サービス共通ライブラリまたはアプリ共通ライブラリの開発の作業効率を向上させることができるという効果を奏する。

【０１２９】また、請求項１１にかかる画像形成装置は、請求項６～１０のいずれか一つに記載の発明において、サービス共通ライブラリがスレッドに関する処理を実行する関数を含んだことで、スレッドに関する関数を共通化してコントロールサービスのアプリケーションの開発の作業効率を向上させることができるという効果を奏する。

【０１３０】また、請求項１２にかかるアドレス解決方法は、呼び出す関数のメモリ上での絶対アドレスを、関

数配置情報に基づいて、アプリケーションおよびコントロールサービスのアドレス参照変換部に格納するアドレス参照解決ステップを含んだことで、実行時のアドレス解決の処理が簡易なものとなり、プログラム実行時のオーバヘッドを短縮することができるという効果を奏する。

【０１３１】また、請求項１３にかかるアドレス解決方法は、請求項１２に記載の発明において、アドレス参照解決ステップがサービス共通ライブラリまたはアプリ共通ライブラリの先頭の絶対アドレスと該先頭の絶対アドレスからの関数のオフセットをアドレス参照変換部に格納することで、共通ライブラリやアプリ共通ライブラリの先頭の絶対アドレスを定めておけば、関数配置情報には先頭アドレスからのオフセットを格納しておけばよく、これによってサービス共通ライブラリまたはアプリ共通ライブラリの開発の作業効率を向上させることができるという効果を奏する。

【図面の簡単な説明】

【図１】この発明の実施の形態によるプログラム作成装置のハードウエア構成を示す説明図である。

【図２】この発明の実施の形態によるプログラム作成装置の構成を機能的に示す説明図である。

【図３】この発明の実施の形態による実行プログラム記憶部２００ａに記憶される共有プログラムファイル３００の一例を模式的に示す説明図である。

【図４】この発明の実施の形態によるアドレス参照変換モジュール作成部２０１により作成され、アドレス参照変換モジュール記憶部２００ｂに格納されたアドレス参照変換モジュールの一例を模式的に示す説明図である。

【図５】この発明の実施の形態によるプログラム変換部２０２により機械語に変換され、プログラム結合部２０３により必要な関数が結合された後、実行プログラム記憶部２００ａに格納されたプログラムの一例を模式的に示す説明図である。

【図６】この発明の実施の形態によるプログラム作成装置の、プログラムの作成から実行までの処理の流れを示すフローチャートである。

【図７】実施の形態２にかかる複合機の構成を示すブロック図である。

【図８】実施の形態２にかかる複合機におけるコントロールサービスとサービス共通ライブラリとランタイムローダの関係を示す模式図である。

【図９】実施の形態２にかかる複合機のＳＣＳのプロセス起動時における仮想メモリ上の配置を示す模式図である。

【図１０】実施の形態２にかかる複合機におけるサービス共通ライブラリの構造を示す模式図である。

【図１１】実施の形態２にかかる複合機におけるＳＣＳの本体プログラムの内容を示す説明図である。

【図１２】実施の形態２の複合機におけるランタイムロ

ーダのアドレス解決処理の手順を示すフローチャートである。

【図１３】実施の形態２の複合機のプリンタアプリのプロセス起動時における仮想メモリ上の配置を示す模式図である。

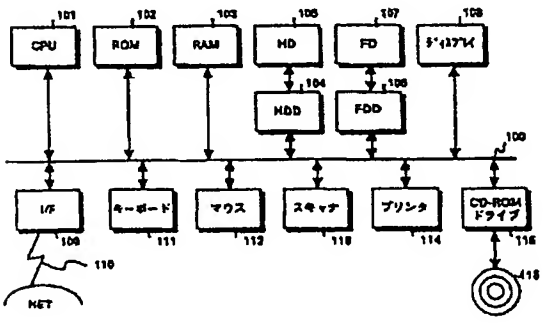
【図１４】実施の形態３にかかる複合機におけるアプリ共通ライブラリの構造を示す模式図である。

【図１５】実施の形態３の複合機におけるプリンタアプリの本体プログラムの内容を示す説明図である。

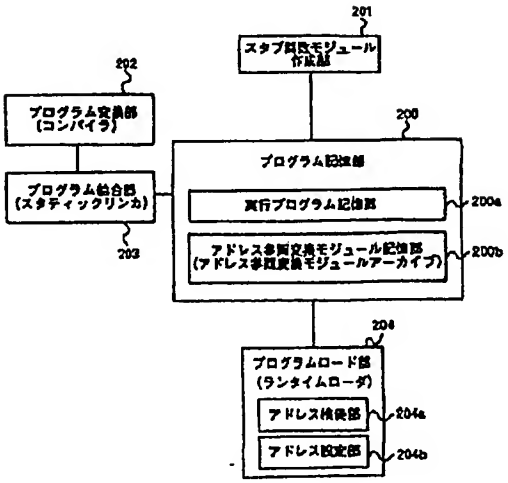
【符号の説明】

１００　バスまたはケーブル
１０１　ＣＰＵ
１０２　ＲＯＭ
１０３　ＲＡＭ
１０４　ＨＤＤ
１０５　ＨＤ
１０６　ＦＤＤ
１０７　ＦＤ
１０８　ディスプレイ
１０９　Ｉ／Ｆ
１１０　通信回線
１１１　キーボード
１１２　マウス
１１３　スキャナ
１１４　プリンタ
１１５　ＣＤ－ＲＯＭ
１１６　ＣＤ－ＲＯＭドライブ
２００　プログラム記憶部
２００ａ　実行プログラム記憶部
２００ｂ　アドレス参照変換モジュール記憶部
２０１　アドレス参照変換モジュール作成部
＊２０２　プログラム変換部
２０３　プログラム結合部
２０４　プログラムロード部
２０４ａ　アドレス検索部
２０４ｂ　アドレス設定部
７００　複合機
７０１　白黒ラインプリンタ
７０２　カラーラインプリンタ
７０３　ハードウェアリソース
７１０　ソフトウェア群
７１１　プリンタアプリ
７１２　コピーアプリ
７１３　ファックスアプリ
７１４　スキャナアプリ
７１５　ネットファイルアプリ
７１６　工程検査アプリ
７２０　プラットホーム
７２１　汎用ＯＳ
７２２　ＳＣＳ
７２３　ＳＲＭ
７２４　ＥＣＳ
７２５　ＭＣＳ
７２６　ＯＣＳ
７２７　ＦＣＳ
７２８　ＮＣＳ
７３０　アプリケーション
８４０　サービス共通ライブラリ
８４１　アプリ共通ライブラリ
８４２　アドレス参照変換部
８４３　ランタイムローダ
＊８４４　関数配置情報

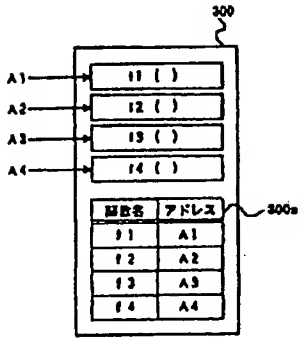
【図１】



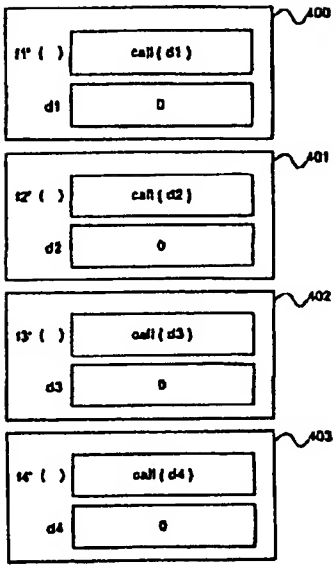
【図２】



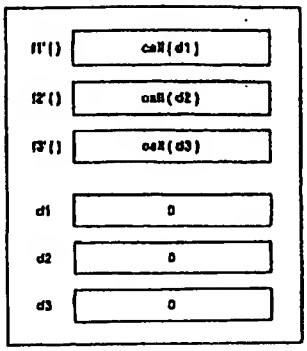
【図３】


300
A1
A2
A3
A4
f1 ( )
f2 ( )
f3 ( )
f4 ( )
関数名
アドレス
f1
A1
f2
A2
f3
A3
f4
A4
300a

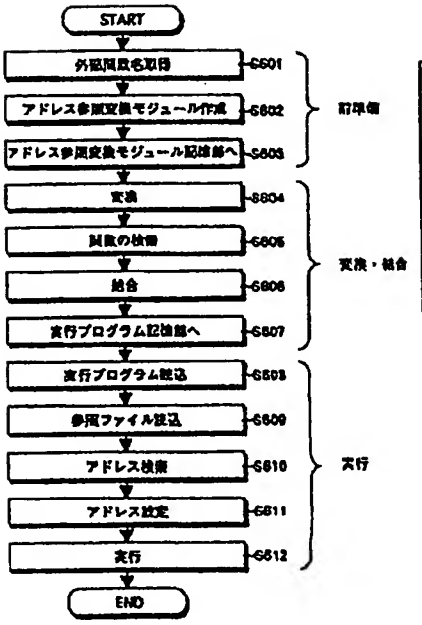
【図４】


400
f1' ( )
call(d1)
d1
0
401
f2' ( )
call(d2)
d2
0
402
f3' ( )
call(d3)
d3
0
403
f4' ( )
call(d4)
d4
0

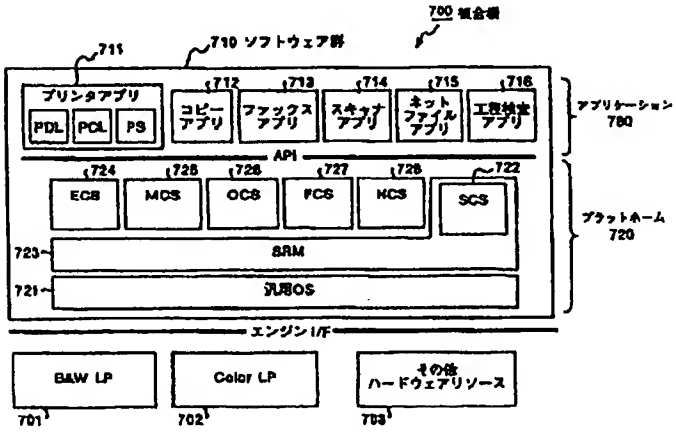
【図５】


f1'()
call(d1)
f2'()
call(d2)
f3'()
call(d3)
d1
0
d2
0
d3
0

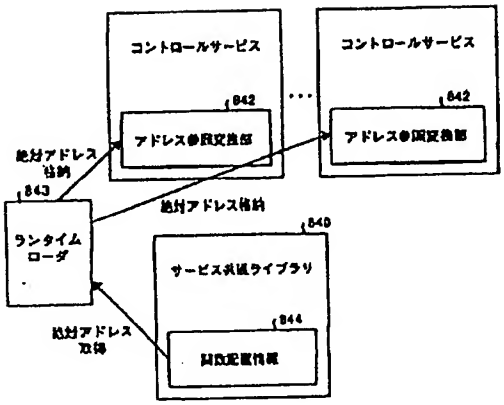
【図６】


START
外部関数名取得
S601
アドレス参照変換モジュール作成
S602
アドレス参照変換モジュール記憶部へ
S603
前準備
変換
S604
関数の検索
S605
結合
S606
実行プログラム記憶部へ
S607
変換・結合
実行プログラム読込
S608
参照ファイル読込
S609
アドレス検索
S610
アドレス設定
S611
実行
S612
実行
END

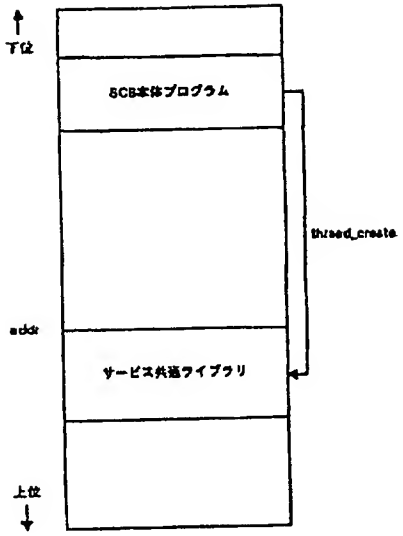
【図７】


700 複合機
711
710 ソフトウェア群
プリンタアプリ
PDL
PCL
PS
712
コピーアプリ
713
ファックスアプリ
714
スキャナアプリ
715
ネットファイルアプリ
716
工程検査アプリ
アプリケーション 730
API
724
ECS
725
MCS
726
OCS
727
FCS
728
NCS
722
SCS
723
SRM
721
汎用OS
プラットホーム 720
エンジンI/F
B&W LP
701
Color LP
702
その他ハードウェアリソース
703

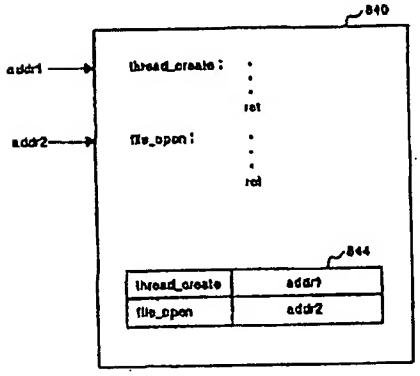
【図８】



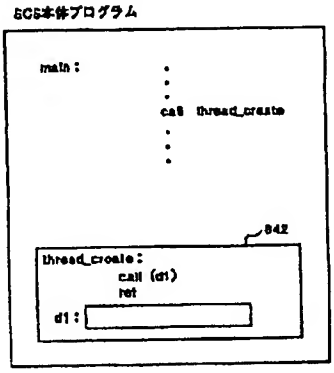
【図９】



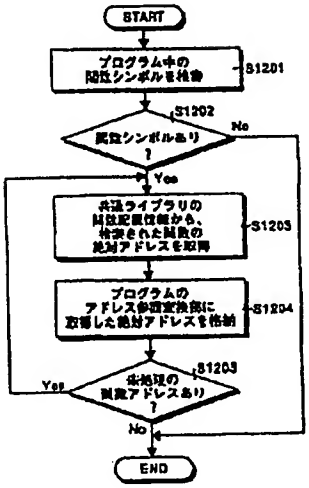
【図１０】



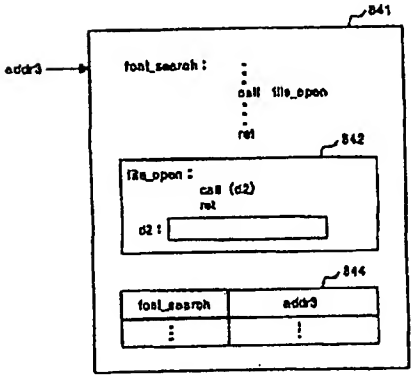
【図１１】



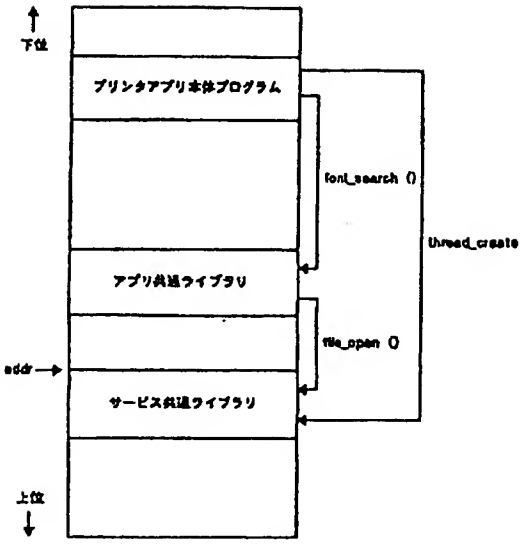
【図１２】



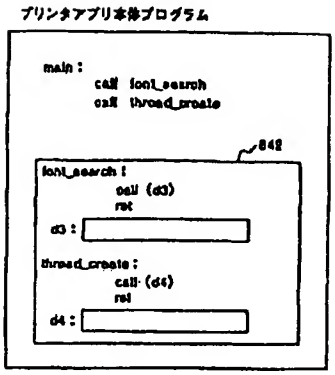
【図１４】



【図１３】



【図１５】



(11) Japanese Patent Laid-Open No. 2002-222081

(43) Laid-Open Date: August 9, 2002

(21) Application No. 2001-280642

(22) Application Date: September 14, 2001

(71) Applicant: Ricoh Co., Ltd.

(72) Inventor: Yoshiaki IRINO

(74) Agent: Patent Attorney, Hiroaki SAKAI

(54) [Title of the Invention] DEVICE AND METHOD FOR PROGRAM GENERATION, PROGRAM IMPLEMENTING THE SAME METHOD ON COMPUTER, IMAGE FORMING DEVICE, AND ADDRESS SOLVING METHOD

(57) [Abstract]

[Object] To improve the working efficiency and to reduce the size of a program by making it easy to develop respective modules individually.

[Solving Means] A service common library 840 includes one or more functions called in common from a plurality of control services and function arrangement information 844 showing the absolute memory addresses of the respective functions. Each control service has an address reference conversion part 842 where the absolute address of a function called by a run-time loader 843 is stored when the function is executed.

[Claims]

[Claim 1] A program preparation device for preparing a program to execute a program by a computer comprising:
a preparation means for preparing an address reference conversion module consisting of a jump instruction to an external function module referred to by the program;
a conversion means for converting the program into a computer-executable format; and
a linking means for linking an address reference conversion module prepared by the preparation means and consisting of the jump instruction to the external function module referred to by the program to the program converted by the conversion means.

[Claim 2] The program preparation device according to Claim 1 comprising:
a retrieval means for retrieving the address of the external function module of the jump destination of the address reference conversion module linked by the linking means during the execution of the program; and
a setting means for setting the address retrieved by the retrieval means to the address reference conversion module linked by the linking means.

[Claim 3] A program preparation method for preparing a program to be executed by a computer including:
a preparation step of preparing an address reference

conversion module consisting of a jump instruction to an external function module referred to by the program;
a conversion step of converting the program into a computer-executable format; and
a linking step of linking the address reference conversion module prepared in the preparation step and consisting of the jump instruction to the external function module referred to by the program to the program converted by the conversion step.

[Claim 4] The program preparation method according to Claim 3 including:
a retrieval step of retrieving the address of the external function module of a jump destination of the address reference conversion module linked by the linking step during the execution of the program; and
a setting step of setting the address retrieved in the retrieval step to the address reference conversion module linked by the linking step.

[Claim 5] A program for executing the method according to Claim 3 or Claim 4 by a computer.

[Claim 6] An image forming device comprising: hardware resources to be used in the image forming processing; an application for performing the processing specific to each of the user services engaged in the image forming processing; and a platform which is interposed between the

application and the hardware resources and including a plurality of control services for performing the acquisition request, the control, the execution control and the image forming processing of the hardware resources commonly requested by at least two applications when providing the user services; and

further comprising: an application common library including one or a plurality of functions to be commonly called from the plurality of applications and function arrangement information indicating the absolute memory address of each function,

wherein the applications have an address reference conversion part with the absolute address of the function to be called being stored during the execution.

[Claim 7] The image forming device according to Claim 6, further comprising a service common library having one or a plurality of functions to be commonly called from the plurality of control services, and function arrangement information indicating the absolute memory address on the memory of each function,

wherein the control services have the address reference conversion part with the absolute address of the function to be called being stored during the execution.

[Claim 8] The image forming device according to Claim 7 wherein the service common library further comprises one or

a plurality of functions to be called from the application common library and function arrangement information indicating the absolute memory address on the memory of each function, and

wherein the application common library has an address reference conversion part with the absolute address of the function to be called being stored during the execution.

[Claim 9] The image forming device according to any one of Claims 6 to 8, further comprising an address reference solution means for storing the absolute address of the function to be called in the address reference conversion part based on the function arrangement information when starting the image forming device.

[Claim 10] The image forming device according to Claim 9, wherein the address reference solution means stores the leading absolute address of the service common library or the application common library and the offset of the function from the leading absolute address in the address reference conversion part.

[Claim 11] The image forming device according to any one of Claims 6 to 10, wherein the control services and the application are operated as the process including one or a plurality of threads; and

wherein the service common library includes the function for executing the processing of the thread.

[Claim 12] An address solution method for solving the address of the function to be called from the control services and the application when starting an image forming device comprising: hardware resources to be used in the image forming processing; an application for performing the processing specific to each of the user services engaged in the image forming processing; a platform which is interposed between the application and the hardware resources and includes a plurality of control services for performing the acquisition request, the control, the execution control and the image forming processing of the hardware resources commonly requested by at least two applications when providing the user services; an application common library including one or a plurality of functions to be commonly called from the plurality of applications and function arrangement information indicating absolute memory address on the memory of each function; and a service common library having one or a plurality of functions to be commonly called from the plurality of control services and function arrangement information indicating the absolute memory address of each function; and

including an address reference solution step of storing the absolute memory address on the memory of the function to be called in the address reference conversion part of the application and the control services based on the function

arrangement information.

[Claim 13] The address solution method according to Claim 12, wherein the address reference solution step stores the leading absolute address of the service common library or the application common library and the offset of the function from the leading absolute address in the address reference conversion part.

[Detailed Description of the Invention]

[0001]

[Technical Field of the Invention] The present invention relates to a program preparation device and a program preparation method for preparing a program to be executed by a computer, a program for executing the method by the computer, and an address solution method of a function to be generated by linking a control service and an application to be operated with a common library and called from an image forming device, the control service, and the application when providing the user services related to the image forming processing such as a printer, a copier or a facsimile.

[0002]

[Description of the Related Art] In recent years, an image forming device in which the function of each device such as a printer, a copier, a facsimile, and a scanner is stored in one casing (hereinafter, referred to as a "compound

machine") has been generally known. The compound machine has a display unit, a printing unit and an image pickup unit or the like in one casing, and also has three kinds of software respectively corresponding to the printer, the copier and the facsimile device, and operates the compound machine as the printer, the copier, the scanner or the facsimile device by changing the software.

[0003] In such a conventional compound machine, the software (including a versatile OS) corresponding to the printer, the copier, the scanner and the facsimile device is separately provided therein, and a very long time is required for developing each software. In addition, the application software operated in such the compound machine has been generally developed at one department of a development agent of the compound machine.

[0004] Thus, the applicant invented an image forming device having a platform prepared of various kinds of control services which has hardware resources to be used in the image forming processing by the display unit, the printing unit and the image pickup unit, and mounts a plurality of applications to perform the processing specific to each user service of the printer, the copier or the facsimile, and is interposed between these applications and the hardware resources to perform the control, the execution control and the image forming processing of the hardware resources

commonly requested by at least two applications when providing the user services. This image forming device has the platform which performs the control, the execution control and the image forming processing of the hardware resources commonly requested by at least two applications, the efficiency of the software development can be ensured, and the productivity as the entire device can be enhanced.

[0005] When developing the software to be operated by such the compound machine, it is preferable for enhancing the development working efficiency and reducing the size of the program by sharing the external functions or the like to be commonly called from a plurality of modules as a common library, and linking a module with the external function modules to complete the module when generating an object file from a source file of the module to be developed.

[0006] For a method for linking the module to be developed with the external function modules, (1) a static link (linking before execution) system and (2) a dynamic link (linking at execution) system described below have been generally known.

[0007] (1) Static link (linking before execution) system: This is a system in which all the possibly necessary modules are linked with each other in advance. In this static link system, the absolute address on the virtual memory of the function in the common library is directly referred to on

the side of the module to be developed. Therefore, it is advantageous that the necessary module can be rapidly executed as necessary.

[0008] (2) Dynamic link (linking at execution) system: This is a system for linking the module requested during the execution as needed. The same module can be referred to from a plurality of programs. Thus, it is advantageous that the size of the entire program can be reduced by preparing the library of the module to be used in common.

[0009]

[Problems to be Solved by the Invention] However, both (1) the static link (linking before execution) system and (2) the dynamic link (linking at execution) system include the following technical problems in the above-described advantages when developing the software to be operated by the compound machine.

[0010] Firstly, the static link (linking before execution) system has a problem in that the size of the entire program is increased since all modules are linked with the execution program in advance.

[0011] Further, in the above-described compound machine, each module has an independent constitution having a plurality of control services providing services which are commonly requested by at least two of a plurality of applications and applications, and the development is

preferably and independently performed for each application or each control service. For example, there can be a development method in that the module of the control service is developed by the development agent of the compound machine while the development of the application is committed to a third party such as a third vender.

[0012] However, when the static link system is employed, the development must be performed so that the absolute address on the virtual memory of the external function must be directly referred to on the module side of the object to be developed, and a problem occurs, in that each module is hardly prepared separately. In particular, when upgrading the compound machine by changing the specification of the external function of the common library or adding a new external function, the absolute address of the function must be changed not only for the common library but also each application to call the common library and the module of each control service, and a problem occurs, in that the working efficiency is poor.

[0013] Further, the dynamic link (linking at execution) system has an overhead associated with the link of the module, and problems occur, in that the performance of the execution program is easily degraded, and that, since the module is linked during the execution, the processing for the address solution must be assembled for each module, and

the size of the module of the dynamic link system is increased in comparison with the module of the static link system.

[0014] In this point, for example, in Japanese Unexamined Patent Application Publication No. 6-250828, the module to be linked is classified according to the level of the frequency of use, and the execution speed of the program is enhanced by performing the static link to the execution program in advance for the module group of high frequency of use, and performing the dynamic link during the execution for the module group of low frequency of use.

[0015] However, in particular, in the embedded execution program, the processing content is changed according to the incoming event from the outside, and a problem occurs, in that it is difficult to estimate the frequency of use of the module. Further, irrespective of the above, the problem is not solved yet, in that the size of the module to be linked during the execution is increased.

[0016] Therefore, in order to solve the problems by the conventional technology, an object of the present invention is to provide a program preparation device, a program preparation method, a program to execute the method by the computer, an image forming device and an address solution method which are capable of easily and independently developing an embedded execution program of the image

forming device for each module, enhancing the working efficiency, reducing the size of the program, and flexibly performing the upgrading and correction of each module.

[0017]

[Means for Solving the Problems] In order to solve the above-described problems and to achieve the object, a program preparation device according to the invention of Claim 1 for preparing a computer-executable program comprises a preparation means for preparing an the address reference conversion module consisting of the jump instruction to an external function module referred to by the program, a conversion means for converting the program into a computer-executable format, and a linking means for linking an address reference conversion module consisting of the jump instruction to the external function module referred to by the program prepared by the preparation means to the program converted by the conversion means.

[0018] In accordance with the invention according to Claim 1, the external function module referred to by the program is executed via the address reference conversion module consisting of the jump instruction to the module subjected to the static link to the program in place of the module.

[0019] Further, the program preparation device according to the invention of Claim 2 comprises a retrieval means for retrieving the address of the external function module of

the jump destination of the address reference conversion module linked by the linking means during the execution of the program in the invention according to Claim 1, and a setting means for setting the address retrieved by the retrieval means to the address reference conversion module linked by the linking means.

[0020] According to the invention of Claim 2, the address of the jump destination of the address reference conversion module is identified during the execution of the program with the module being subjected to the static link.

[0021] Further, a program preparation method according to the invention of Claim 3 for preparing a computer-executable program includes a preparation step of preparing an address reference conversion module consisting of the jump instruction to an external function module referred to by the program, a conversion step of converting the program into a computer-executable format, and a linking step of linking an address reference conversion module consisting of the jump instruction to an external function module referred to by the program and prepared in the preparation step to the program converted by the conversion step.

[0022] According to the invention of Claim 3, the external function module referred to by the program is executed via the address reference conversion module consisting of the jump instruction to the module and subjected to the static

link to the program in place of the module.

[0023] Further, the program preparation method according to the invention of Claim 4 includes, in the invention according to Claim 3, includes a retrieval step for retrieving the address of the external function module of the jump destination of the address reference conversion module linked to the linking step, and a setting step for setting the address retrieved in the retrieval step to the address reference conversion module linked in the linking step.

[0024] According to the invention of Claim 4, the address of the jump destination of the address reference conversion module is identified during the execution of the program with the module subjected to the static link.

[0025] Further, the invention according to Claim 5 is a program to execute the method according to Claim 3 or Claim 4 by a computer, and any one operation of Claim 3 or Claim 4 can be executed by the computer.

[0026] Further, an image forming device according to the invention of Claim 6 comprises hardware resources to be used in the image forming processing, an application for performing the processing specific to each of the user services engaged in the image forming processing, and a platform which is interposed between the application and the hardware resources and including a plurality of control

services for performing the acquisition request, the control, the execution control and the image forming processing of the hardware resources commonly requested by at least two applications when providing the user services, and further comprises one or a plurality of functions to be commonly called from the plurality of applications, and the application common library including function arrangement information indicating the absolute memory address on the memory of each function, wherein the applications has an address reference conversion part with the absolute address of the function to be called is stored during the execution.

[0027] The invention of Claim 6 has one or a plurality of functions to be commonly called from a plurality of applications, and the application common library including function arrangement information indicating the absolute memory address on the memory of each function, and the application has the address reference conversion part in which the absolute address of the function to be called is stored during the execution, and if the application has the address reference conversion part, any development conscious of the absolute address of the function to be called is not requested, and the application can be easily developed separately from the application common library or other applications. Further, the development of the application need not be determined when developing the absolute address

of the function to be called by providing the address reference conversion part, and the version upgrade of the application can be easily performed. Therefore, the working efficiency of development of the application can be enhanced.

[0028] The invention according to Claim 7 further comprises, in the image forming device according to Claim 6, a service common library having one or a plurality of functions to be commonly called from the plurality of control services, and function arrangement information indicating the absolute memory address on the memory of each function, and the control service has the address reference conversion part in which the absolute address of the function to be called is stored during the execution.

[0029] The invention according to Claim 7 further comprises a service common library having one or a plurality of functions to be commonly called from a plurality of control services, and function arrangement information indicating the absolute memory address on the memory of each function, and the control service has the address reference conversion part in which the absolute address of the function to be called is stored during the execution. Thus, any development conscious of the absolute address of the function to be called need not be performed by providing the address reference conversion part in the control services, and the development of the control services can be easily

performed separately from the service common library or other control services. Further, the development of the control services need not be determined when developing the absolute address of the function to be called by providing the address reference conversion part, and the version upgrade of the control services can be easily performed. Therefore, the working efficiency of development of the control services can be enhanced.

[0030] Further, in the invention according to Claim 8, the service common library has one or a plurality of functions to be called from the application common library and function arrangement information of the absolute memory address on the memory of each function in the image forming device according to Claim 7, and the application common library has the address reference conversion part in which the absolute address of the function to be called is stored during the execution.

[0031] In the invention according to Claim 8, the service common library further comprises one or a plurality of functions to be called from the application common library and function arrangement information indicating the absolute memory address on the memory of each function, and the application common library has the address reference conversion part in which the absolute address of the function to be called is stored during the execution. Thus,

any development conscious of the absolute address of the function to be called need not be performed by providing the address reference conversion part on the application common library, and the application common library can be easily performed separately from the service common library. Further, the development of the application common library need not be determined when developing the absolute address of the function to be called by providing the address reference conversion part, and the version upgrade of the application common library can be easily performed. Therefore, the working efficiency of development of the application common library can be enhanced.

[0032] Further, the invention according to Claim 9 further comprises an address reference solution means for storing the absolute address of the function to be called in the address reference conversion part based on the function arrangement information when starting the image forming device in the image forming device according to any one of Claims 6 to 8.

[0033] In the invention according to Claim 9, the absolute address of the function to be called is stored in the address reference conversion part by the address reference solution means based on function arrangement information when starting the image forming device, the address solution can be easily processed during the execution, and the

overhead when executing the program can be shortened.

[0034] In the invention according to Claim 10, the address reference solution means stores the leading absolute address of the service common library or the application common library and the offset of the function from the leading absolute address in the address reference conversion part in the image forming device according to Claim 9.

[0035] In the invention according to Claim 10, the address reference solution means stores the leading absolute address of the service common library or the application common library and the offset of the function from the leading absolute address in the address reference conversion part, and by determining the leading absolute address of the common library or the application common library, only the offset from the leading address may be stored in function arrangement information, and the working efficiency of development of the service common library or the application common library can be enhanced.

[0036] In the invention according to Claim 11, the control services and the application are operated as the process including one or a plurality of threads, and the service common library includes the function for executing the processing on the thread in the image forming device according to any one of Claims 6 to 10.

[0037] In the invention according to Claim 11, the service

common library includes the function for executing the processing on the thread, and the working efficiency of development of the application of the control services can be enhanced by communizing the function on the thread.

[0038] In the invention according to Claim 12, an address solution method for solving the address of the function to be called from the control services and the application includes an address reference solution step of storing the absolute memory address of the function to be called in the address reference conversion part of the application and the control services based on the function arrangement information when starting an image forming device comprising hardware resources to be used in the image forming processing, an application for performing the processing specific to each of the user services engaged in the image forming processing, a platform which is interposed between the application and the hardware resources and includes a plurality of control services for performing the acquisition request, the control, the execution control and the image forming processing of the hardware resources commonly requested by at least two applications when providing the user services, an application common library including one or a plurality of functions to be commonly called from the plurality of applications and function arrangement information indicating absolute memory address on the memory

of each function, and a service common library having one or a plurality of functions to be commonly called from the plurality of control services and function arrangement information indicating the absolute memory address on the memory of each function.

[0039] The invention according to Claim 12 includes an address reference solution step of storing the absolute memory address of the function to be called in the address reference conversion part of the application and the control services based on the function arrangement information, and the processing of the address solution during the execution is simply performed, and the overhead when executing the program can be shortened.

[0040] In the invention according to Claim 13, the address reference solution step stores the leading absolute address of the service common library or the application common library and the offset of the function from the leading absolute address in the address reference conversion part in the address solution method according to Claim 12.

[0041] In the invention according to Claim 13, the address reference solution step stores the leading absolute address of the service common library or the application common library and the offset of the function from the leading absolute address is stored in the address reference conversion part, and by determining the leading absolute

address of the common library and the application common library, the offset from the leading address may only be stored in function arrangement information, and the working efficiency of development of the service common library or the application common library can be enhanced.

[0042]

[Embodiments] Hereinafter, preferable embodiments of a program preparation device, a program preparation method, a program to execute the method by the computer, an image forming device and an address solution method in accordance with the present invention will be described in detail with reference to the attached drawings.

[0043] (Embodiment 1)

(Hardware constitution of the program preparation device)

Firstly, the hardware constitution of the program preparation device according to the embodiment of the present invention will be described. Fig. 1 is an explanatory view showing the hardware constitution of the program preparation device according to the embodiment of the present invention.

[0044] In the figure, reference numeral 101 denotes a CPU for controlling the entire device, reference numeral 102 denotes a ROM storing a basic input-output program, and reference numeral 103 denotes a RAM to be used for a work area of the CPU 101, respectively.

[0045] Further, reference numeral 104 denotes a HDD (Hard Disk Drive) for controlling the read/write of the data from/in a HD (Hard Disk) 105 following the control of the CPU 101, and reference numeral 105 denotes a HD for storing the data written following the control of the HDD 104, respectively.

[0046] Still further, reference numeral 106 denotes an FDD (Floppy Disk Drive) for controlling the read/write of the data from/in a FD (Floppy (registered trademark) Disk) 107 following the control of the CPU 101, and reference numeral 107 denotes an attachable/detachable FD for storing the data written following the control of the FDD 106, respectively.

[0047] Still further, reference numeral 108 denotes a display for displaying various kinds of data such as a cursor, a menu, a window and characters and images, reference numeral 109 denotes a network board which is connected to a network NET via a communication line 110 and functions as an interface between the network and the CPU 101, respectively. Still further, reference numeral 111 denotes a keyboard having a plurality of keys for inputting characters, numerals, various kinds of instructions or the like, and reference numeral 112 denotes a mouse for performing the selection and execution of various kinds of instructions, selection of an object to be processed, movement of the cursor, or the like, respectively.

[0048] Still further, reference numeral 113 denotes a scanner for optically reading characters and images, reference numeral 114 denotes a printer for printing the characters and images following the control of the CPU 101, reference numeral 115 denotes a CD-ROM which is an attachable/detachable recording medium, reference numeral 116 denotes a CD-ROM drive for controlling the control of the read of the data from the CD-ROM 115, and reference numeral 100 denotes a bus or a cable for connecting the above-described components to each other.

[0049] (Functional constitution of the program preparation device) Next, the functional constitution of the program preparation device according to the embodiment of the present invention will be described. Fig. 2 is an explanatory view which functionally shows the constitution of the program preparation device according to the embodiment of the present invention. The program preparation device according to the embodiment of the present invention includes the program storage unit 200, the address reference conversion module preparation unit 201, the program conversion unit 202, the program linking unit 203, and the program load unit 204.

[0050] Reference numeral 200 denotes a program storage unit, which includes an execution program storage unit 200a and an address reference conversion module storage unit 200b.

Firstly, the execution program storage unit 200a stores an executable program in a classified and arranged manner in a plurality of files. The program file stored therein roughly includes (a) a file in which a module group to be commonly referred to from a plurality of other program files is described, in other words, a common program file and (b) a file describing a program for referring the modules via a below-described address reference conversion module.

[0051] Fig. 3 is an explanatory view schematically showing an example of the common program file 300 to be stored in the execution program storage unit 200a. The external function modules f1, f2, f3 and f4 which are called by other programs and used are subjected to the static link in advance to this common file, and stored in the addresses A1, A2, A3 and A4, respectively. Further, a symbol table 300a with the name and the address of the external function module included therein corresponding to each other is stored in each common file.

[0052] Further, the address reference conversion module storage unit (an address reference conversion module archive) 200b stores an address reference conversion module prepared by a below-described address reference conversion module preparation unit 201. The address reference conversion module will be described immediately below.

[0053] Next, reference numeral 201 denotes an address

reference conversion module preparation unit, which prepares an address reference conversion module corresponding to each external function module in the common program file stored in the execution program storage unit 200a. The prepared address reference conversion module is stored in the address reference conversion module storage unit 200b.

[0054] Fig. 4 is an explanatory view schematically showing an example of the address reference conversion module prepared by the address reference conversion module preparation unit 201 and stored in the address reference conversion module storage unit 200b. For example, reference numeral 400 denotes an address reference conversion module corresponding to the external function f1 shown in Fig. 3, in which the address reference conversion f1' having the same name as that of the external function f1 and the data area d1 with the address value when executing the external function f1 being stored are defined. This function f1' performs only the processing of "jump to the address stored in the data area d1".

[0055] Further, the address value of zero, in other words, the impossible address value is stored in the data area d1 at this time, and as described below, the real address value of the external function f1 is set by the program load unit 204 when executing the program. Further, the address reference conversion modules 401 to 403 corresponding to the

external functions f2 to f4 have the same format, which can be prepared automatically if the function name is known.

[0056] Reference numeral 202 denotes a program conversion unit (a compiler), which converts a source program input from the keyboard 111 or read out from the HD 105 into an object program executable by the computer.

[0057] Reference numeral 203 denotes a program linking unit (a static linker), which retrieves the address reference conversion module corresponding to the external function referred to by the program, in other words, the address reference conversion module having the function of the same name as that of the external function from the address reference conversion module storage unit 200b, and links it to the program. The executable program file with the necessary functions being linked thereto is stored in the execution program storage unit 200a.

[0058] Fig. 5 is an explanatory view schematically showing an example of a program stored in the execution program storage unit 200a after being converted in the machine language by the program conversion unit 202 with the requested function being linked thereto by the program linking unit 203. When it is assumed that this program originally refers to the external functions f1, f2 and f3, the corresponding address reference conversion modules f1', f2' and f3' are linked to the program after the conversion

and linkage in place of these external functions. The values of the data areas of d1, d2 and d3 are not set yet.

[0059] Next, reference numeral 204 denotes a program load unit (a run-time loader), which includes an address retrieval unit 204a and an address setting unit 204b. The program load unit 204 starts the program stored in the execution program storage unit 200a. Here, a case in which the program shown in Fig. 5 is started will be described as an example.

[0060] Firstly, the program load unit 204 retrieves the program file referred to by the program, i.e., a common program file 300 shown in Fig. 3 from the execution program storage unit 200a. The symbol table 300a of this common program file 300 will be retrieved by the address retrieval unit 204a to acquire the address value of the external function corresponding to the function having the same name as that of the address reference conversion linked to the above-described program, in other words, the address reference conversion. In this example, the address values A1, A2 and A3 of the external functions f1, f2 and f3 corresponding to the address reference conversion modules f1', f2' and f3' will be acquired.

[0061] Next, the program load unit 204 stores the address values in the data area d1, d2 and d3 of the program by the address setting unit 204b. The jump destinations by the

address reference conversion modules f1', f2' and f3' are determined, and the preparation of executing the program is prepared. When executing the program, the corresponding external function will be executed via the linked address reference conversion module.

[0062] Further, the address reference conversion module preparation unit 201, the program conversion unit 202, the program linking unit 203 and the program load unit 204 realize the function of each unit by the execution of the command processing by the CPU 101 according to the command of the program read by the RAM 103 from various kinds of recording media such as the HD 105 and the FD 107 or the CD-ROM 115.

[0063] (Processing flow from preparation to execution of program) Next, the processing flow from the preparation to the execution of the program in the program preparation device according to the embodiment of the present invention will be described. Fig. 6 is a flowchart indicating the processing flow of the preparation to the execution of the program of the program preparation device according to the embodiment of the present invention.

[0064] Further, the processing in (1) to (3) is not necessarily performed continuously, but there may be a temporal interval between the respective processing. However, in the figure, the entire flow is shown.

[0065] (1) In the advanced preparation Steps S601 to S603, firstly, the address reference conversion module corresponding to each external function module is prepared in the address reference conversion module preparation unit 201. In Step S601, the address reference conversion module preparation unit 201 retrieves the symbol table of the common program file 300 stored in the execution program storage unit 200a, and successively acquires the name of the external function described in the file.

[0066] In the succeeding Step S602, the address reference conversion module consisting of the function of the same name as the external function composed of the jump instruction to each external function, and the data area storing the address of the jump destination of the function will be prepared. The prepared address reference conversion module is stored in the address reference conversion module storage unit 200b in Step S603.

[0067] (2) Conversion and Linkage Thereafter, when the program requiring the conversion and the linkage is input in this device, the program conversion unit 202 converts the input source code into the object code in Step S604 (compile). And, in Step S605, the program linking unit 203 retrieves the function having the same name as that of the external function referred to by the program from the address reference conversion module storage unit 200b, and

links the address reference conversion module including the retrieved function to the program in Step S606 (link). The program is converted into the machine language, and the necessary address reference conversion module is linked thereto. This program is stored in the execution program storage unit 200a in Step S607.

[0068] (3) Execution Thereafter, when the execution instruction of the program is input in this device, the program load unit 204 reads the program instructed from the execution program storage unit 200a in Step S608. And, in Step S609, firstly, the common program file 300 referred to by the program is read from the execution program storage unit 200a.

[0069] In addition, in Step S610, the address retrieval unit 204a of the program load unit 204 retrieves the symbol table of the file retrieved in Step S609, and successively acquires the address of the external function having the same name as that of the address reference conversion linked to the program. And, the address is set in the address reference conversion module of the program by the address setting unit 204b in Step S611. The program to be completely executable by identifying this address is executed in Step S612.

[0070] According to the above-described embodiment, not only the common program file including the entity of the

external function but also the program referring to the external function via the address reference conversion module is completely subjected to the static link, and any command code for enabling the re-arrangement of the object is not included at all, the program size becomes smaller than that in a case of the dynamic link system.

[0071] Further, different from a conventional static link system, the program file including the entity of the external function can be referred to from a plurality of programs, the external function group to be commonly used by a plurality of programs is summarized as the common library, and the size of the entire program can be reduced thereby.

[0072] In addition, any entity of the function is not included in the program referring to the external function, and even in a case in which the inside of the external function is corrected by a reason such as a trouble, only the common program file may be updated, and the development efficiency of the program is enhanced.

[0073] In addition, the static link system of the present invention can be realized without making any change to the compiler or the static linker as a software development tool, and can be coexistent with the conventional development environment, and the above-described advantage can be enjoyed without degrading the efficiency of the software development.

[0074] (Embodiment 2) Fig. 7 is a block diagram showing a constitution of an image forming device according to Embodiment 2 of the present invention (hereinafter, referred to as a "compound machine"). As shown in Fig. 7, the compound machine 700 comprises a black-and-white line printer (B&W LP) 701, a color line printer (Color LP) 702, a hardware resource 703 such as a scanner and a facsimile, and a software application 710 composed of a platform 720 and an application 730.

[0075] The platform 720 has a control service which interprets the processing request from the application and generates an acquisition request of the hardware resources, a system resource manager (SRM) 723 which controls one or a plurality of hardware resources and adjusts the acquisition request form the control service, and the versatile OS 721.

[0076] The control service is formed of a plurality of service modules and comprises an SCS (a system control service) 722, an ECS (an engine control service) 724, an MCS (a memory control service) 725, an OCS (an operation panel control service) 726, an FCS (a facsimile control service) 727, and an NCS (a network control service) 728. The platform 720 has an application program interface (API) capable of receiving the processing request from the application 730 by the function defined in advance.

[0077] The versatile OS 721 is a versatile operating system

such as UNIX (registered trademark), and performs the parallel execution of each software of the platform 720 and the application 730 as the process.

[0078] Each control service is operated as the process, and performs one or a plurality of threads in the process.

[0079] The process of the SRM 723 performs the system control and the resource control together with the SCS 722, and performs the adjustment and the execution control according to the request from an upper rank level using the hardware resources engines such as a scanner unit and a printer unit, a memory, an HDD file, a host I/O (a centro I/F, a network I/F, an IEEE1394 I/F, an RS232C I/F or the like).

[0080] More specifically, the SRM 723 determines whether or not the requested hardware resources are available (whether or not they are not used by other request), and if they are available, it is conveyed to the upper rank level that the requested hardware resources are available. Further, the SRM 723 performs the use scheduling of the hardware resources to the request from the upper rank level, and directly performs the requested content (such as the paper conveyance and the image forming operation by the printer engine, securement of the memory, and generation of the file).

[0081] The process of the SCS 722 performs the control of

the application, the control of the operation unit, the display of the system screen, LED display, resource control, and the control of the interrupted application.

[0082] The process of the ECS 724 controls the engine of the hardware resource 703 controls the engine comprising the black-and-white line printer (B&W LP) 701, the color line printer (Color LP)702, the scanner, the facsimile or the like.

[0083] The process of the MCS 725 acquires and releases the image memory, the use of the hard disk device (HDD), and the compression and enlargement of the image data.

[0084] The process of the OCS 726 controls an operation panel (an operation panel) forming an information transmission means between an operator and a body control.

[0085] The process of the FCS 727 provides the APPEARANCE for performing the transmission-reception of the facsimile using a PSTN/ISDN network from each application layer of the system controller, the registration/citation of various kinds of facsimile data controlled by a BKM (back-up SRAM), the reading of the facsimile, the reception and printing, and of the facsimile, and integrated transmission-reception.

[0086] The NCS 728 is a process for providing the service to be commonly used by the applications requesting the network I/O, and performs the intermediation when sorting the data received by each protocol from the network side to

each application or transmitting the data from the applications to the network side.

[0087] The application 730 performs the processing specific to each user service regarding the image forming such as the printer, the copier, the scanner or the facsimile. The application 730 comprises the printer application 711 as the application for the printer having the page description language (PDL), the PCL and the postscript (PS), the copier application 712 as the application for the copier, the facsimile application 713 as the application for the facsimile, the scanner application 714 as the application for the scanner, the net file application 715 as the application for the network file, and the step inspection application 716 as the application for the process inspection.

[0088] Fig. 8 is a schematic view showing the relationship between the control service, the service common library and the run-time loader in the compound machine according to Embodiment 2. The service common library 840 summarizes a plurality of functions to be commonly called from a plurality of control services such as the ECS 724, the MCS 725, and the SCS 722 in one file so as to be used from the plurality of control services. The function for performing the processing on the processes such as generation, opening and closing of the process, the function for performing the

processing on the thread such as generation, opening and closing of the thread, and the function of the file operation such as the opening and closing of the file, or the like are registered in this service common library 840. Further, function arrangement information 844 indicating the registered absolute address of the function is stored in the service common library 840.

[0089] The control service has an area storing the absolute address of the function to be called during the execution, and the address reference conversion part describing the command of jumping to the absolute address stored in the area.

[0090] The run-time loader 843 acquires each absolute address of the function from function arrangement information 844 of the service common library 840 when executing the application 730 and the system control service, and performs the address solution by storing the absolute address of the function acquired in the address reference conversion part 842 of each control service. This run-time loader 843 constitutes the address reference solution means of the present invention.

[0091] Fig. 10 is a schematic view showing the structure of the service common library 840 in the compound machine according to Embodiment 2. This service common library 840 has an entity part of each function, and function

arrangement information 844 indicating the absolute address in the virtual memory space of each function. In Fig. 10, the thread_create function and the file_open function are registered in the service common library 840 as an example of the functions.

[0092] Function arrangement information 844 of the service common library 840 stores addr1 as the absolute address of the thread_create function, and addr2 as the absolute address of the file_open function. Further, arrangement information of the thread_create function, and arrangement information of the file_open function may be stored as addr0 + offset1, and addr0 + offset2, respectively, where addr0 is the leading absolute address of the service common library 840, offset1 is the offset from the head of the service common library 840 of the thread_create function, and offset2 is the offset from the head of the service common library 840 of the file_open function, respectively.

[0093] The application common library 841 summarizes a plurality of functions to be commonly called from a plurality of applications such as the printer application 711 and the copier application 710 in one file so as to be used from the plurality of applications. The function for performing the processing on the process, the function for performing the processing on the thread, the function for the file operation, or the like are also registered in this

application common library 841. The application common library 841 also has the entity of each function and function arrangement information 844, and its structure is similar to that of the service common library 840.

[0094] Next, the operation when executing the control service in the compound machine according to Embodiment 2 of this constitution will be described with the SCS 722 as an example. Fig. 9 is a schematic view showing the arrangement of the SCS 722 on the virtual memory when the process is started. As shown in Fig. 9, the body program of the SCS 722 is arranged on the lower rank of the address, and the service common library 840 is arranged on the upper rank of the address.

[0095] Fig. 11 is an explanatory view showing the content of the body program of the SCS 722. Further, Fig. 11 mainly shows the function call of the service common library 840.

[0096] As shown in Fig. 11, the body program of the SCS 722 calls the thread_create function of the service common library 840. And, the body program of the SCS 722 has the address reference conversion part 842 for the function call. The data area d1 for storing the absolute address of the thread_create function is ensured in this address reference conversion part 842, and the processing for calling the function of the absolute address to be stored in d1 is described.

[0097] Fig. 12 is a flowchart indicating the procedure of the address solution processing of the run-time loader 843 in the compound machine according to Embodiment 2. When the SCS 722 is started, the run-time loader 843 retrieves the function symbol defined as the external function from the body program of the SCS 722 (Step S1201). Thus, the thread_create function is retrieved, and its symbol thread_create is obtained as the result of retrieval. At this time, the address which is not present in reality (for example, 0x00000000) is stored in the data area d1 of the address reference conversion part 842 of the body program of the SCS 722, and the address solution is not performed yet.

[0098] Then, presence/absence of the function symbol of the external function is determined (Step S1202), and if the function symbol is present, function arrangement information 844 of the service common library 840 is referred to, and the absolute address of the function of the retrieved function symbol is acquired from function arrangement information 844 (Step S1203).

[0099] Next, the acquired absolute address is stored in the data area d1 of the corresponding function of the address reference conversion part 842 of the body program (Step S1204). Thus, the absolute address addr1 is stored in the data area d1 to perform the address solution of the thread_create function.

[0100] By repeating the processing of the above-described Steps S1203 and S1204 to every function symbol retrieved in Step S1201 (Step S1205), the address solution processing of the external function of the body program of the SCS 722 is completed.

[0101] And, when the address solution of every function is completed, the execution of the main of the body program of the SCS 722 is started. When the thread_create function is called to generate the thread during the execution of the main, the control is shifted to the absolute address addr1 of the thread_create function by the address reference conversion part 842, and the entity of the thread_create function of the service common library 840 is called to generate the thread.

[0102] In Embodiment 2, the execution of the SCS 722 is described. However, similar processing is performed on the execution of other control services or applications. During the execution of the applications, however, the address solution processing is performed with the functions in the application common library 841.

[0103] As described above, the compound machine according to Embodiment 2 has the service common library 840 having the entity of the function to be commonly called from a plurality of control services and function arrangement information 844 indicating the absolute address on the

virtual memory of each function. Since each control service has the address reference conversion part 842 in which the absolute address of the function is stored by the run-time loader 843 during the execution, any development need not be performed while being conscious of the absolute address of the function, and the development of the control services can be easily performed independently from the common library and other control services. Further, in the development of the control services, the absolute address of the function to be called need not be determined during the development by providing the address reference conversion part 842, the version upgrade of the control services can be easily performed, and the working efficiency of development of the control services can be enhanced.

[0104] Further, in the compound machine according to Embodiment 2, the absolute address of the function is stored in the address reference conversion part 842 based on function arrangement information 844 when starting the control service by the run-time loader 843. Thus, the processing of the address solution during the execution becomes simple, the overhead when executing the program is reduced.

[0105] (Embodiment 3) The compound machine according to Embodiment 2 has the service common library 840 registering the function called from a plurality of control services and

the application common library 841 registering the function called from a plurality of applications, and the common libraries are independent from each other. However, the compound machine according to Embodiment 3 calls the function further registered in the service common library 840 from the functions registered in the application common library 841.

[0106] The constitution of the compound machine according to Embodiment 3 is similar to that of the compound machine according to Embodiment 2, and its description will be omitted. Fig. 14 is a schematic view showing a structure of the application common library 841 in the compound machine according to Embodiment 3. This application common library 841 has an entity part of each function and function arrangement information 844 indicating the absolute address in the virtual memory space of each function. In Fig. 14, the font_search function is registered in the application common library 841 as an example of the function, and has function arrangement information 844 indicating the absolute address addr3 of the font_search function. In addition, as shown in Fig. 14, the file_open function of the service common library 840 is called from the font_search function of the application common library 841. Thus, the application common library 841 has the address reference conversion part 842 for calling the file_open function.

This address reference conversion part 842 ensures the data area d2 storing the absolute address of the file_open function, and describes the processing for calling the function of the absolute address to be stored in d2.

[0107] Next, the operation during the execution of the application in the compound machine according to Embodiment 3 of this constitution will be described with the printer application 711 as an example. Fig. 13 is a schematic view showing the arrangement of the printer application 711 on the virtual memory when starting the process. As shown in Fig. 13, the body program of the printer application 711 is arranged on the lower rank of the address, the application common library 841 is arranged on its upper rank, and the service common library 840 is further arranged on its upper rank. Further, the address at which the service common library 840 is arranged is fixed to the same address as that of the service common library 840 according to Embodiment 2. In other words, each library is arranged at the position fixed by the absolute address.

[0108] Fig. 15 is an explanatory view indicating the content of the body program of the printer application 711. Fig. 15 mainly shows the function call of the application common library 841 and the service common library.

[0109] As shown in Fig. 15, the body program of the printer 711 calls the font_search function of the application common

library 841and the thread_create function of the service common library 840. And, the body program of the printer application 711 has the address reference conversion part 842 for these function calls. This address reference conversion part 842 ensures the data area d3 storing the absolute address of the font_search function, and describes the processing for calling the function of the absolute address to be stored in the data area d3. Further, the address reference conversion part 842 ensures the data area d4 storing the absolute address of the thread_create function, and describes the processing for calling the function of the absolute address stored in d4. Further, before starting the printer application 711, the address which is not present in reality (for example, 0x00000000) is stored in the data areas d3 and d4, and the address is in a non-solved state.

[0110] When the body program of the printer application 711 is started, the run-time loader 843 performs the address solution processing of the function. In other words, the function symbol of the external function to be called from the application common library 841 is retrieved. The absolute address of the retrieved function symbol file_open function is acquired with reference to function arrangement information 844 of the service common library 840. The absolute address addr2 is thus acquired, and the absolute

address addr2 is stored in the data area d2 of the address reference conversion part 842 in the application common library 841.

[0111] Next, the run-time loader 843 performs the address solution processing in the body program of the printer application 711. In other words, the function symbol of the external function to be called from the body program of the printer application 711 is retrieved. In this condition, the font_search and the thread_create are retrieved as the function symbols.

[0112] And, the run-time loader 843 acquires the absolute address of the function of the retrieved function symbol font_search with reference to function arrangement information 844 of the application common library 841. Thus, the absolute address addr3 is acquired, and this absolute address addr3 is stored in the data area d3 the address reference conversion part 842 in the body program of the printer application 711.

[0113] Next, the run-time loader 843 refers to function arrangement information 844 of the application common library 841 in order to obtain the absolute address of the function of another retrieved function symbol thread_create. Any thread_create is not registered in function arrangement information 844 of the application common library 841, and then, function arrangement information 844 of the service

common library 840 is referred to. The absolute address addr1 of the thread_create is acquired from function arrangement information 844 of the service common library 840, and the absolute address addr1 is stored in the data area d4 of the address reference conversion part 842 of the body program of the printer application 711. Thus, every address solution processing is completed.

[0114] When every address solution is completed, the execution of the main of the body program of the printer application 711 is started. When the font_search function for retrieving the font during the execution of the main is called, the control is shifted to the absolute address addr3 of the font_search function by the address reference conversion part 842, and the entity of the font_search function of the application common library 841 is called, and executed.

[0115] When the file_open function for the file open is called while executing the font_search function, the control is shifted to the absolute address addr2 of the file_open function by the address reference conversion part 842, and the entity of the file_open function of the service common library 840 is called, and executed.

[0116] Next, when the thread_create function is called to generate the thread by the body program of the printer application 711, the control is shifted to the absolute

address addr1 of the thread_create function by the address reference conversion part 842, and the entity of the thread_create function of the service common library 840 is called and executed.

[0117] In Embodiment 3, the execution of the printer application 711 is described, and similar processing is performed for the execution of other applications.

[0118] Thus, in the compound machine according to Embodiment 3, the service common library 840 has the function to be called from the application common library 841 and function arrangement information 844 indicating the absolute memory address of each function, and the application common library 841 has the address reference conversion part 842 in which the absolute address of the function to be called is stored during the execution. Thus, any development conscious of the absolute address of the function to be called need not be performed, and the independent development of the application common library 841 from that of the service common library 840 can be easily performed. Further, the development of the application common library 841 need not be determined during the development of the absolute address of the function by providing the address reference conversion part 842, the version upgrade of the application common library 841 is easily performed, and the working efficiency of development

of the application common library 841 can be enhanced.

[0119]

[Advantages] As described above, in the program preparation device according to Claim 1 of the present invention, the program preparation device for preparing the program to be executed by the computer comprises a preparation means for preparing the address reference conversion module consisting of the jump instruction to the external function module with the program referring thereto, a conversion means for converting the program in a computer-executable format, and a linking means for linking the address reference conversion module consisting of the jump instruction to the external function module with the program referring thereto which is prepared by the preparation means to the program converted by the conversion means; and the external function module to be referred to from the program is executed via the address reference conversion module consisting of the jump instruction to the module subjected to the static link to the program in place of the module. Thus, there is an advantage of obtaining the program preparation device capable of suppressing the size of the embedded execution program, and flexibly adaptive to the upgrade and correction of each module.

[0120] Further, the program preparation device according to Claim 2 of the present invention further comprises a

retrieval means for retrieving the address of the external function module of the jump destination of the address reference conversion module linked by the linking means during the execution of the program, and a setting means for setting the address retrieved by the retrieval means to the address reference conversion module linked by the linking means, and the address of the jump destination of the address reference conversion module is identified during the execution of the program with the module being subjected to the static link, demonstrating the advantage that the program preparation device flexibly adaptive to the upgrade and correction of each module to be assembled to the program can be obtained for the embedded execution program.

[0121] Further, in the program preparation method according to Claim 3 of the present invention, the program preparation method for preparing the program to be executed by the computer includes a preparation step of preparing an address reference conversion module consisting of the jump instruction to the external function module with the program referring thereto, a conversion step of converting the program in a computer-executable format, and a linking step of linking the address reference conversion module consisting of the jump instruction to the external function module with the program referring thereto prepared in the preparation step to the program converted in the conversion

step, and the external function module referred to from the program is executed via the address reference conversion module consisting of the jump instruction to the module subjected to the static link to the program in place of the module, demonstrating an advantage of obtaining the program preparation method capable of suppressing the size of the embedded execution program and flexibly adaptive for the update and correction of each module.

[0122] Further, the program preparation method according to Claim 4 of the present invention, in the invention according to Claim 3, includes a retrieval step of retrieving the address of the external function module of the jump destination of the address reference conversion module linked in the linking step, and a setting step of setting the address retrieved in the retrieval step to the address reference conversion module linked in the linking step, and the address of the jump destination of the address reference conversion module is identified during the execution of the program subjected to the static link of the module, demonstrating an advantage of obtaining the program preparation method flexibly adaptive to the upgrade and correction of each module corresponding assembled to the program to the embedded execution program.

[0123] The invention according to Claim 5 is a program to execute the method according to Claim 3 or Claim 4 by the

computer, demonstrating an advantage that any one operation of Claim 3 or Claim 4 can be executed by the computer.

[0124] Further, the image forming device according to Claim 6 comprises the application common library including one or a plurality of functions to be commonly called from a plurality of applications, and function arrangement information indicating the absolute memory address of each function, and the application has the address reference conversion part in which the absolute address of the function to be called is stored during the execution. Thus, by providing the address reference conversion part in the application, any development conscious of the absolute address of the function to be called need not be performed, the development of the application can be easily performed separately from the application common library or other applications. Further, the development of the application need not be determined when developing the absolute address of the function to be called by providing the address reference conversion part, and the version upgrade of the application can be easily performed, demonstrating the advantage that the working efficiency of development of the application can be enhanced thereby.

[0125] Further, the image forming device according to Claim 7 further comprises, in the invention according to Claim 6, the service common library having one or a plurality of

functions to be commonly called from a plurality of control services and function arrangement information indicating the absolute memory address on the memory of each function and the absolute memory address of each function. The control services have the address reference conversion part in which the absolute address of the function to be called is stored during the execution, and by providing the address reference conversion part in the control services, any development conscious of the absolute address of the function to be called need not be performed, and the development of the control services can be easily performed separately from the service common library or other control services. Since the development of the control services need not be determined when developing the absolute address of the function to be called by providing the address reference conversion part. The version upgrade of the control services can be easily performed, demonstrating the advantage that the working efficiency of development of the control services can be enhanced.

[0126] Further, the image forming device according to Claim 8 has, in the invention according to Claim 7, the service common library further has one or a plurality of functions to be called from the application common library and function arrangement information indicating the absolute memory address on the memory of each function, and the

application common library has the address reference conversion part in which the absolute address of the function to be called is stored during the execution. Thus, by providing the address reference conversion part in the application common library, any development conscious of the absolute address of the function to be called need not be performed, the development of the application common library can be easily performed separately from the service common library. Further, the development of the application common library need not be determined when developing the absolute address of the function to be called by providing the address reference conversion part, and the version upgrade of the application common library can be easily performed, demonstrating the advantage that the working efficiency of development can be enhanced.

[0127] Further, the image forming device according to Claim 9 has, in any one invention according to Claims 6 to 8, an address reference solution means for storing the absolute address of the function to be called in the address reference conversion part based on function arrangement information when starting the image forming device. The processing of the address solution during the execution becomes simple, demonstrating the advantage that the overhead when executing the program is shortened.

[0128] Further, in the image forming device according to

Claim 10, the address reference solution means stores, in the invention according to Claim 9, the leading absolute address of the service common library or the application common library and the offset of the function from the leading absolute address in the address reference conversion part. By determining the leading absolute address of the service common library and the application common library, the offset from the leading address may only be stored in function arrangement information, demonstrating the advantage that the working efficiency of development of the service common library or the application common library can be enhanced.

[0129] Further, the image forming device according to Claim 11 includes the function of executing the processing of the thread by the service common library according to any one of Claims 6 to 10, demonstrating the advantage that the working efficiency of development of the application of the control services can be enhanced by communizing the function on the thread.

[0130] Further, the address solution method according to Claim 12 includes the address reference solution step of storing the absolute memory address of the function to be called in the address reference conversion part of the application and the control services based on function arrangement information. Thus, the processing of the

address solution during the execution becomes simple, demonstrating the advantage that the overhead when executing the program can be shortened.

[0131] Further, in the address solution method according to Claim 13, the address reference solution step stores, in the invention according to Claim 12, the leading absolute address of the service common library or the application common library and the offset of the function from the leading absolute address in the address reference conversion part. By determining the leading absolute address of the common library and the application common library, the offset from the leading address may only be stored in function arrangement information, demonstrating the advantage that the working efficiency of development of the service common library or the application common library can be enhanced.

[Brief Description of the Drawings]

[Fig. 1] Fig. 1 is an explanatory view showing the hardware constitution of a program preparation device according to an embodiment of the present invention.

[Fig. 2] Fig. 2 is an explanatory view functionally showing the constitution of the program preparation device according to the embodiment of the present invention.

[Fig. 3] Fig. 3 is an explanatory view schematically showing an example of a common program file 300 stored in an

execution program storage unit 200a according to the embodiment of the present invention.

[Fig. 4] Fig. 4 is an explanatory view schematically showing an example of an address reference conversion module prepared by an address reference conversion module preparation unit 201 and stored in an address reference conversion module storage unit 200b according to the embodiment of the present invention.

[Fig. 5] Fig. 5 is an explanatory view schematically showing a program which is converted into a machine language by a program conversion unit 202 according to the embodiment of the present invention, and stored in the execution program storage unit 200a after necessary functions are linked by a program linking unit 203.

[Fig. 6] Fig. 6 is a flowchart indicating the processing flow from the preparation to the execution of the program of the program preparation device according to the embodiment of the present invention.

[Fig. 7] Fig. 7 is a block diagram showing the constitution of a compound machine according to Embodiment 2.

[Fig. 8] Fig. 8 is a schematic view showing the relationship between the control services, the service common library and the run-time loader in the compound machine according to Embodiment 2.

[Fig. 9] Fig. 9 is a schematic view showing the arrangement

on a virtual memory when starting the SCS process of the compound machine according to Embodiment 2.

[Fig. 10] Fig. 10 is a schematic view showing a structure of a service common library in the compound machine according to Embodiment 2.

[Fig. 11] Fig. 11 is an explanatory view indicating the content of a body program of SCS in the compound machine according to Embodiment 2.

[Fig. 12] Fig. 12 is a flowchart indicating a procedure of the address solution processing of the run-time loader in the compound machine according to Embodiment 2.

[Fig. 13] Fig. 13 is a schematic view showing the arrangement on the virtual memory when starting the process of the printer application of the compound machine according to Embodiment 2.

[Fig. 14] Fig. 14 is a schematic view showing a structure of an application common library in the compound machine according to Embodiment 3.

[Fig. 15] Fig. 15 is an explanatory view showing the content of a body program of a printer application in the compound machine according to Embodiment 3.

[Reference Numerals]

100 bus or cable

101 CPU

102 ROM

103 RAM

104 HDD

105 HD

106 FDD

107 FD

108 display

109 I/F

110 communication line

111 keyboard

112 mouse

113 scanner

114 printer

115 CD-ROM

116 CD-ROM drive

200 program storage unit

200a execution program storage unit

200b address reference conversion module storage unit

201 address reference conversion module preparation unit

202 program conversion unit

203 program linking unit

204 program load unit

204a address retrieval unit

204b address setting unit

700 compound machine

701 black-and-white line printer

702 color line printer

703 hardware resource

710 software application

711 printer application

712 copier application

713 facsimile application

714 scanner application

715 net file application

716 step inspection application

720 platform

721 versatile OS

722 SCS

723 SRM

724 ECS

725 MCS

726 OCS

727 FCS

728 NCS

730 application

840 service common library

841 application common library

842 address reference conversion part

843 run-time loader

844 function arrangement information

Fig. 1

108 display

111 keyboard

112 mouse

113 scanner

114 printer

116 CD-ROM drive

Fig. 2

202 program conversion unit (compiler)

203 program linking unit (static linker)

201 stub function module preparation unit

200 program storage unit

200a execution program storage unit

200b address reference conversion module storage unit (address reference conversion module archive)

204 program loading unit (run-time loader)

204a address retrieval unit

204b address setting unit

Fig. 3

300a function name address

Fig. 6

S601 acquisition of external function name

S602 preparation of address reference conversion module

S603 to address reference conversion module storage unit

S604 conversion

S605 retrieval of function

S606 linkage

S607 to execution program storage unit

S608 read execution program

S609 read reference file

S610 address retrieval

S611 address setting

S612 execution

(1) pre-preparation

(2) conversion and linkage

(3) execution

Fig. 7

700 compound machine

711 printer application

710 software group

712 copy application

713 facsimile application

714 scanner application

715 net file application

716 process inspection application

730 application

720 platform

721 general OS

703 other hardware resource

(1) engine I/F

Fig. 8

842 address reference conversion unit

843 run-time loader

840 service common library

844 function arrangement information

(1) control service

(2) store absolute address

(3) acquire absolute address

(4) store absolute address

(5) control service

Fig. 9

(1) lower rank

(2) upper rank

(3) SCS body program

(4) service common library

Fig. 11

(1) SCS body program

Fig. 12

S1201 retrieve function symbol in program

S1202 is function symbol present?

S1203 acquire absolute address of retrieved function from function arrangement information of common library

S1204 store acquired absolute address in address reference conversion unit of program

S1205 Is non-processed function address present?

Fig. 13

(1) lower rank

(2) upper rank

(3) printer application body program

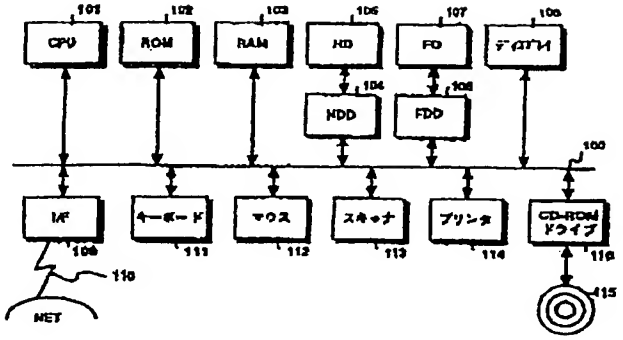
(4) application common library

(5) service common library

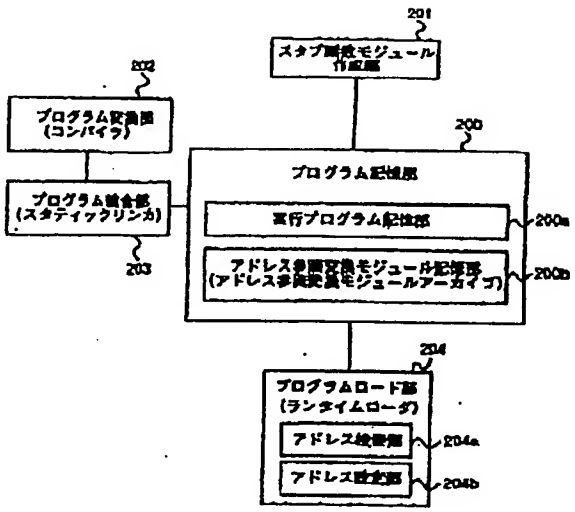
Fig. 15

(1) printer application body program

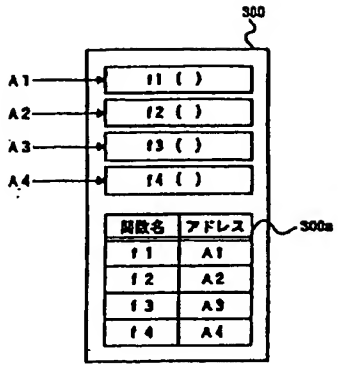
【図１】


CPU
ROM
RAM
HD
FD
HDD
FDD
IF
キーボード
マウス
スキャナ
プリンタ
CD-ROM ドライブ
NET

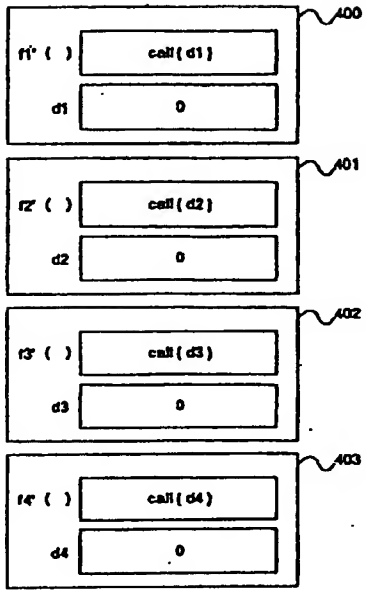
【図２】


スタブ関数モジュール作成部
プログラム変換部（コンパイラ）
プログラム結合部（スタティックリンカ）
プログラム記憶部
実行プログラム記憶部
アドレス参照変換モジュール記憶部（アドレス参照変換モジュールアーカイブ）
プログラムロード部（ランタイムローダ）
アドレス検索部
アドレス設定部

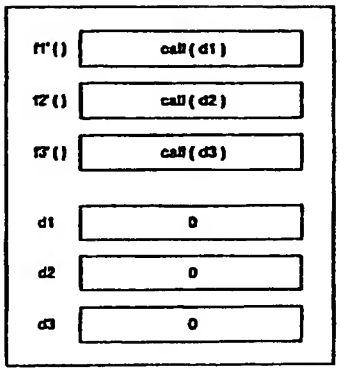
【図3】


300
A1
A2
A3
A4
f1 ( )
f2 ( )
f3 ( )
f4 ( )
関数名
アドレス
f1
A1
f2
A2
f3
A3
f4
A4
300a

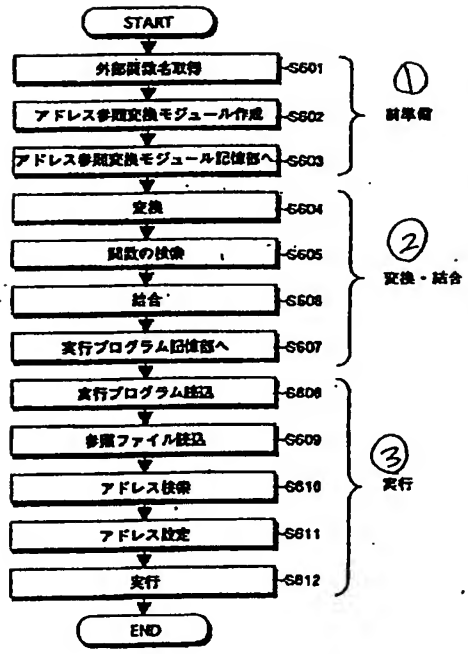
【図4】


400
f1' ( )
call(d1)
d1
0
401
f2' ( )
call(d2)
d2
0
402
f3' ( )
call(d3)
d3
0
403
f4' ( )
call(d4)
d4
0

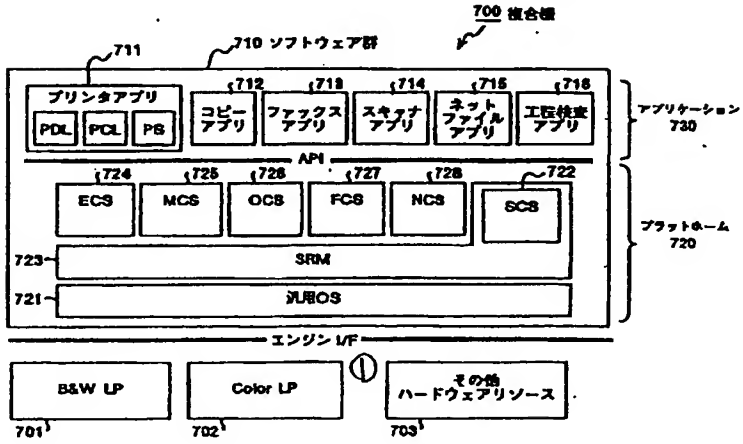
【図5】


f1'()
call(d1)
f2'()
call(d2)
f3'()
call(d3)
d1
0
d2
0
d3
0

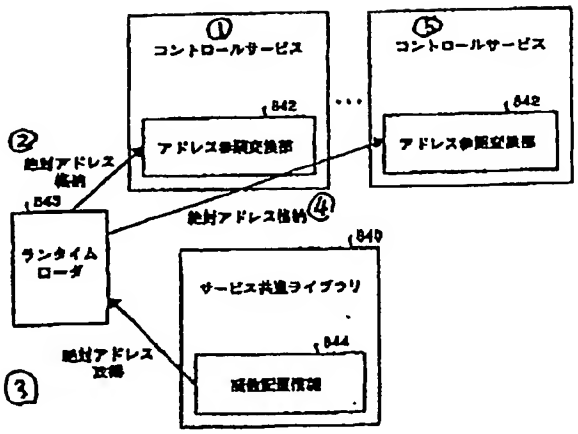
【図6】


START
外部関数名取得
S601
アドレス参照変換モジュール作成
S602
アドレス参照変換モジュール記憶部へ
S603
①
前準備
変換
S604
関数の検索
S605
結合
S606
実行プログラム記憶部へ
S607
②
変換・結合
実行プログラム読込
S608
参照ファイル読込
S609
アドレス検索
S610
アドレス設定
S611
実行
S612
③
実行
END

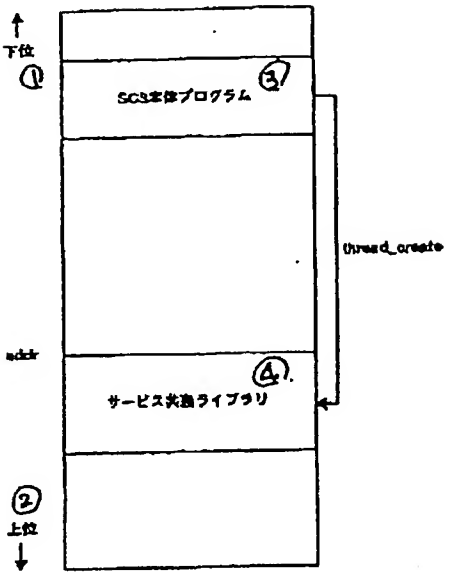
【図7】


700 複合機
711
710 ソフトウェア群
プリンタアプリ
PDL
PCL
PS
712
コピーアプリ
713
ファクスアプリ
714
スキャナアプリ
715
ネットファイルアプリ
716
工程検査アプリ
アプリケーション 720
API
724
ECS
725
MCS
726
OCS
727
FCS
728
NCS
722
SCS
723
SRM
721
汎用OS
プラットホーム 720
エンジンI/F
B&W LP
701
Color LP
702
①
その他ハードウェアリソース
703

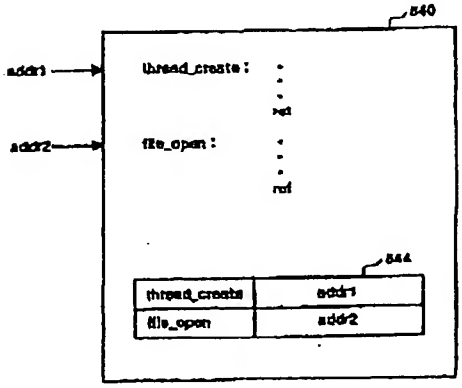
【図８】



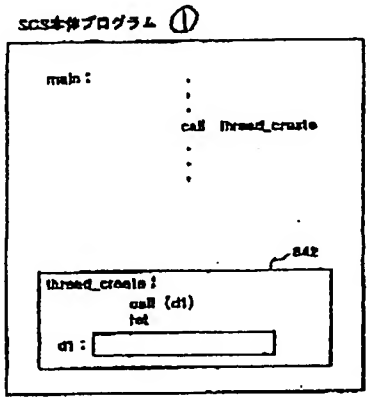
【図９】



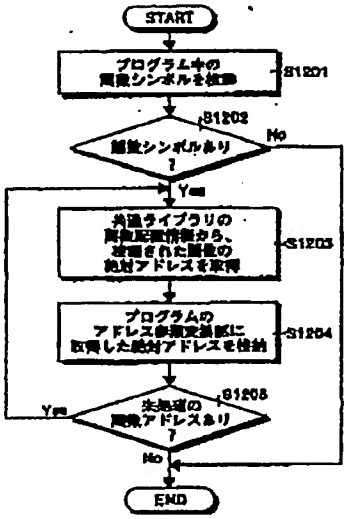
【図１０】



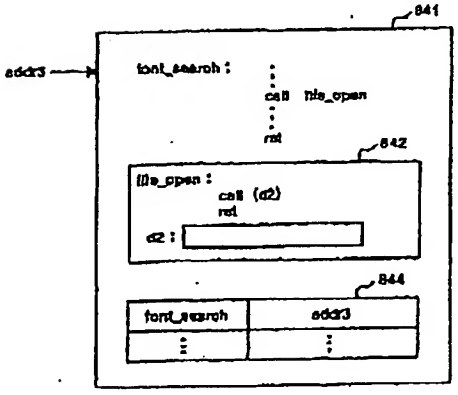
【図１１】



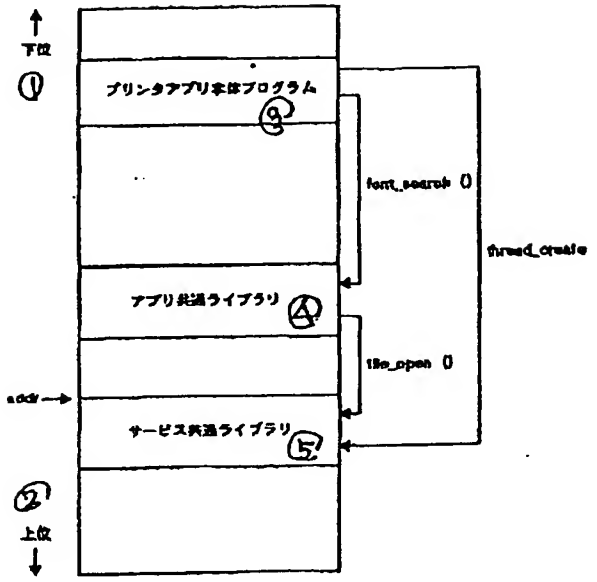
【図１２】



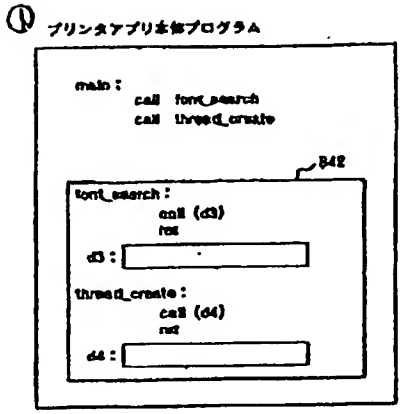
【図１４】



【図１３】


下位
①
プリンタアプリ本体プログラム
font_search ()
thread_create
アプリ共通ライブラリ
file_open ()
addr
サービス共通ライブラリ
②
上位

【図１５】


①
プリンタアプリ本体プログラム
main :
call font_search
call thread_create
342
font_search :
call (d3)
ret
d3 :
thread_create :
call (d4)
ret
d4 :